UNIVERSITÉ DE NANTES

ECOLE **C**ENTRALE DE **N**ANTES

LABORATOIRE D'AUTOMATIQUE DE NANTES

DIPLÔME D'ETUDES APPROFONDIES :
GÉNIE-MÉCANIQUE

OPTION : CMAO-PRODUCTIQUE

De la fabrication, dès la conception

Présenté par : CHABLAT Damien
le 22 Septembre 1995

Jury : F. Bennis
J. Y. Hascoët
F. Le Maitre

Remerciements

Ce travail a été effectué au sein de l'équipe C.M.A.O. et Productique du laboratoire d'Automatique de l'Ecole Centrale de Nantes.

Je tiens à remercier M. Patrick CHEDMAIL de m'avoir accueilli et d'avoir mis à ma disposition les moyens nécessaires à la réalisation de ce travail.

Mes remerciements vont tout particulièrement à M. HASCOET et à M. BENNIS qui m'ont encadré et apporté aide et conseils.

Pour leur accueil et leur patience, je remercie aussi toutes les personnes que j'ai pu contacter dans l'optique de la réalisation de ce travail.

1. TABLE DES MATIÈRES

2. ILLUSTRATIONS

PREMIÈRE PARTIE :

BIBLIOGRAPHIE

3. INTRODUCTION

L'introduction de la C.A.O.[1] et de la C.F.A.O.[2] dans les entreprises, a permis une diminution du temps consacré à la conception, tout en augmentant la qualité et la fiabilité des produits. Cependant, pour répondre aux besoins de la concurrence, l'industrie doit continuer à améliorer ses performances. Pour ce faire, les logiciels doivent pouvoir intégrer dès la conception des produits, toutes les contraintes liées à son cycle de vie, c'est à dire de la mercatique (Besoin du client) à sa destruction (Recyclage), en passant par sa production (Moindre coût). L'objet de ce stage de D.E.A. est d'introduire les contraintes dues à la fabrication dès le stade de la conception.

Pour atteindre ce but, il est nécessaire de connaître la structure actuelle des logiciels et les liens qui unissent la partie conception (C.A.O.) et la partie fabrication (F.A.O.)[3]. A partir de cela, il sera possible d'analyser le rôle des " Features " qui permettent, en plus de la description géométrique et topologique de la pièce, l'introduction de données techniques.

[1] C.A.O. : Conception Assistée par Ordinateur.

[2] C.F.A.O. : Conception et Fabrication Assistée par Ordinateur.

[3] F.A.O. : Fabrication Assistée par Ordinateur.

4. LES MODELEURS GÉOMÉTRIQUES

4.1. INTRODUCTION

L'introduction des systèmes de C.A.O. dans les entreprises s'est faite à la demande des bureaux d'études. De cette demande sont nés des systèmes permettant une modélisation de leurs produits. A cause de la faiblesse des moyens informatiques, les premiers systèmes n'étaient qu'une simple reproduction du travail effectué sur les tables à dessin. La première génération des systèmes de C.A.O. ne permettait qu'une représentation plane (2D)[4]. Elle a permis, cependant d'améliorer l'archivage, la réactualisation en vue des modifications et surtout de créer plus rapidement des plans de définition conformes au dessin d'ensemble. La deuxième génération de modeleur a permis la définition spatiale, en introduisant tout d'abord les modeleurs filaires, puis les modeleurs surfaciques, et volumiques.

4.2. PRINCIPES DES MODÈLES DE SOLIDES UTILISÉS EN C.A.O. [GARDAN]

Quatre approches sont utilisées dans les modeleurs actuels pour définir des entités géométriques. On peut les classifier de la manière suivante :

- le modèle par les limites (modèles par les frontières ou B-Rep) : d'abord de type fil de fer, il est maintenant de type solide dans la plupart des systèmes .
- le modèle par historique (arbre de conception ou C.S.G.) : Plus récent, il est encore souvent limité aux opérations booléennes ;

[4] 2D : 2 dimensions.

- les modèles mathématiques : essentiellement appliqués aux courbes et surfaces ;
- les modèles paramètrés : en général décrits par des programmes, ils sont surtout intéressants dans le cas où l'on peut décrire des éléments en fonction de paramètres. Le paramètrage interactif devient un aspect bien traité dans certains systèmes.

4.2.1. Le modèle par les limites

On pourrait résumer ce modèle en disant que le système conserve la " peau " de l'objet et sait, dans le modèle solide, de quel côté est la matière. Un tel modèle comprend en général des informations géométriques (Coordonnées, équations des faces, etc.), des informations topologiques (Façon dont sont reliées les informations géométriques) et des informations annexes (Couleurs des faces, etc.). Il suffit d'orienter par artifice le sens de parcours des contours limitant les faces pour distinguer intérieur et extérieur, suivant la règle de MOEBIUS.

4.2.2. Le modèle par arbre de construction

Ce modèle est appelé ainsi parce qu'il peut être représenté par arbre, bien qu'un modèle plus général soit de type réseau. En général, on trouve aux feuilles de l'arbre des objets primitifs paramètrables et aux noeuds, des opérations. A chaque noeud correspond un objet, même si celui-ci n'est pas réellement " calculé ". En fait, plutôt que de parler d'arbre de construction, il faudrait parler de " conservation de l'historique ". Il y a une volonté de conserver une information " générique ". Ces modèles ont été introduits, depuis relativement peu de temps dans les systèmes de C.A.O. Ils sont souvent limités aux opérations booléennes et ne prennent pas forcément en compte tous les types d'objets.

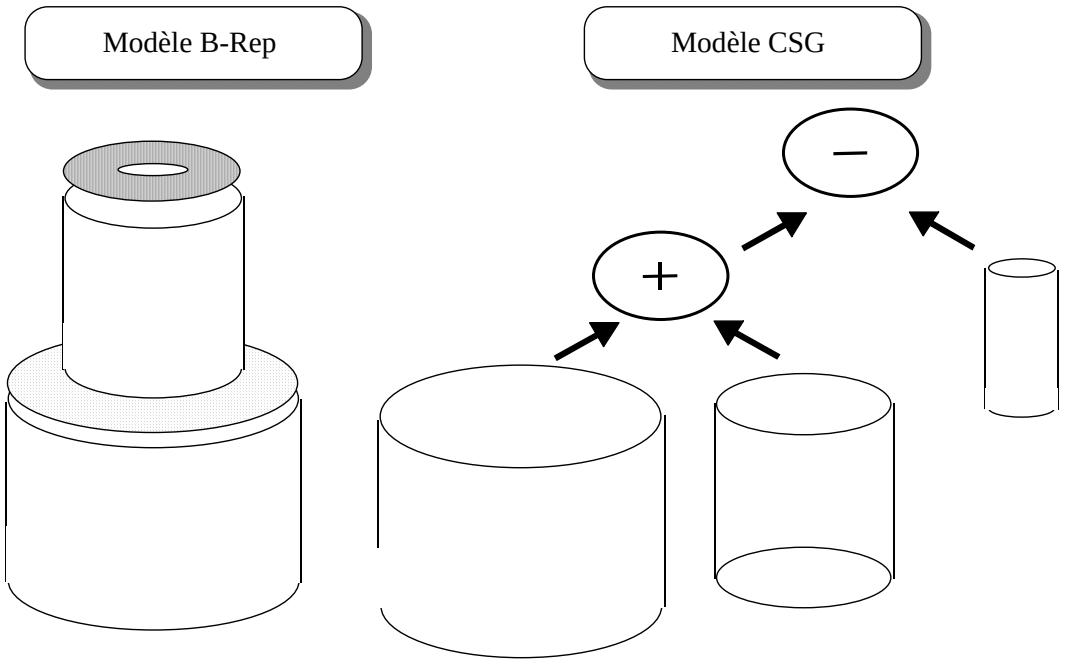


Figure 1 : Modèle C.S.G. & B-Rep

4.2.3. Les modèles mathématiques

Les modèles mathématiques ne sont utilisés que pour les courbes et surfaces. Les surfaces sont, en général, décrites par " morceaux ". Cette définition impose des contraintes, par exemple pour les raccordements.

On utilise dans tous les systèmes des représentations sous forme polynomiale rationnelle ou non. Les trois types de modèles les plus courants dans les systèmes de C.F.A.O. sont :

- les courbes et surfaces de Bézier ;
- les courbes et surfaces B-Splines ;
- les courbes et surfaces N.U.R.B.S.[5]

Chacun de ces modèles est une généralisation de la précédente.

[5] N.U.R.B.S. : Non Uniforme Rational B-Spline.

4.2.4. Les modèles paramètrés

On ne conserve que la façon dont doit être construit l'objet en fonction de certains paramètres. En général, les objets paramètrés sont décrits par des programmes qui sont parfois en mode interactif. Un objet particulier est donc simplement décrit par le programme générateur de toute la famille et par des paramètres définissant cet objet particulier.

4.2.5. Structure d'un modeleur

Un modeleur solide utilise conjointement deux types de modèle, le modèle C.S.G. et B-Rep, chacun de ces deux modèles ayant ses avantages. Le modèle C.S.G. permet la modification de chacune de ses entités de base et le modèle B-Rep permet son affichage à l'écran. Cependant, si la conversion entre le modèle C.S.G. vers le modèle B-Rep est possible, le contraire est actuellement impossible.

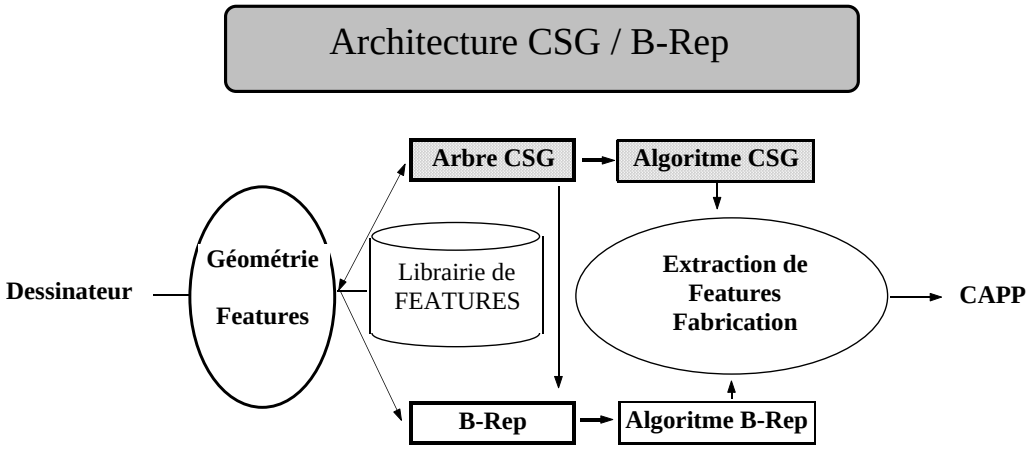


Figure 2 : Architecture des systèmes C.S.G. & B-Rep

Pour supporter les diverses applications incluses dans le concept C.I.M., la structure d'un modeleur doit être ouverte pour lui permettre de dialoguer avec son environnement (Fabrications ,Simulations, Calculs, Plans, etc.).

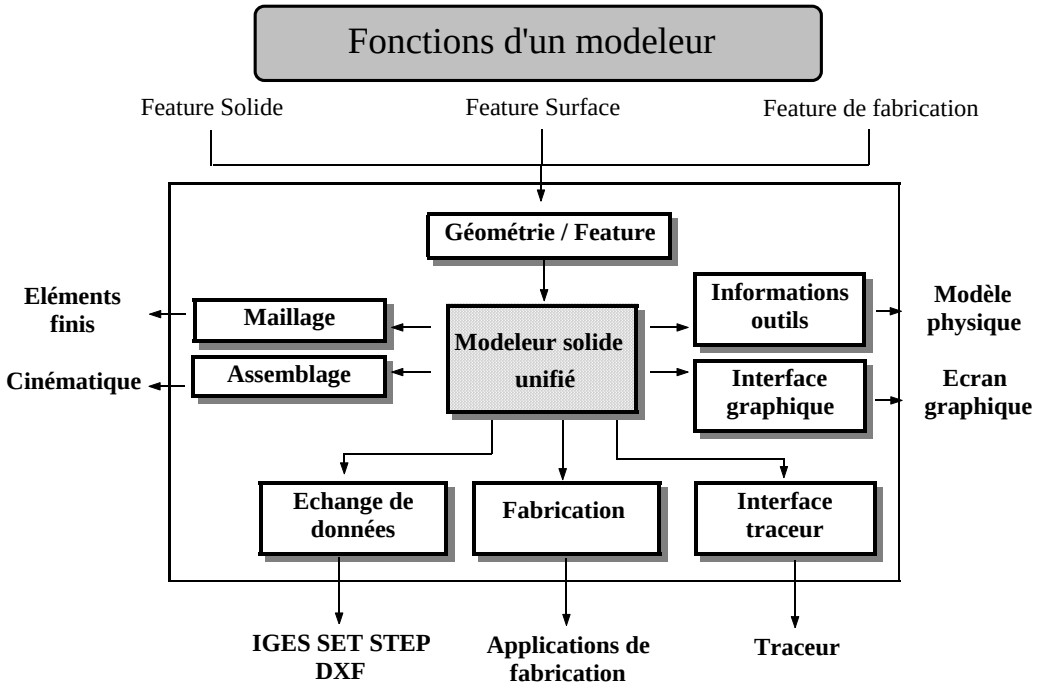


Figure 3 : Fonctions d'un modeleur

2.3. LES " FEATURES " [SCHULTE]

Les " Features ", éléments principaux des systèmes de modélisation basé sur ce principe, ont un haut niveau d'abstraction des formes et entités géométriques qui sont utilisées dans des systèmes de raisonnement et sont relatives à la topologie et à la géométrie des objets fabriqués durant le déroulement de la conception et les activités de fabrication. Le but principal des " Features " est la représentation d'entités géométriques spécifiques d'un produit. Les avantages du travail avec des " Features "sont :

- les " Features " augmentent le niveau d'abstraction des objets avec lesquels le dessinateur doit travailler. Plutôt que de se servir avec des lignes, des points, des arcs et des cercles, le dessinateur peut choisir entre des poches, des bossages ou des perçages ;

- les " Features " apportent une méthodologie flexible pour créer une base de données complète du produit. En sélectionnant le format de la base de données pour interroger ou modifier les " Features ", de nouvelles informations peuvent être ajoutées aux " Features " disponibles. Ceci est particulièrement pratique quand on met à jour les " Features " qui sont en liaison avec la pièce modifiée ;
- Les " Features " peuvent être associées avec un ensemble discret d'options pour la planification de la production, ainsi on sélectionne la méthode optimale de fabrication pour ces " Features ". La sélection est basée sur l'interaction des " Features ".

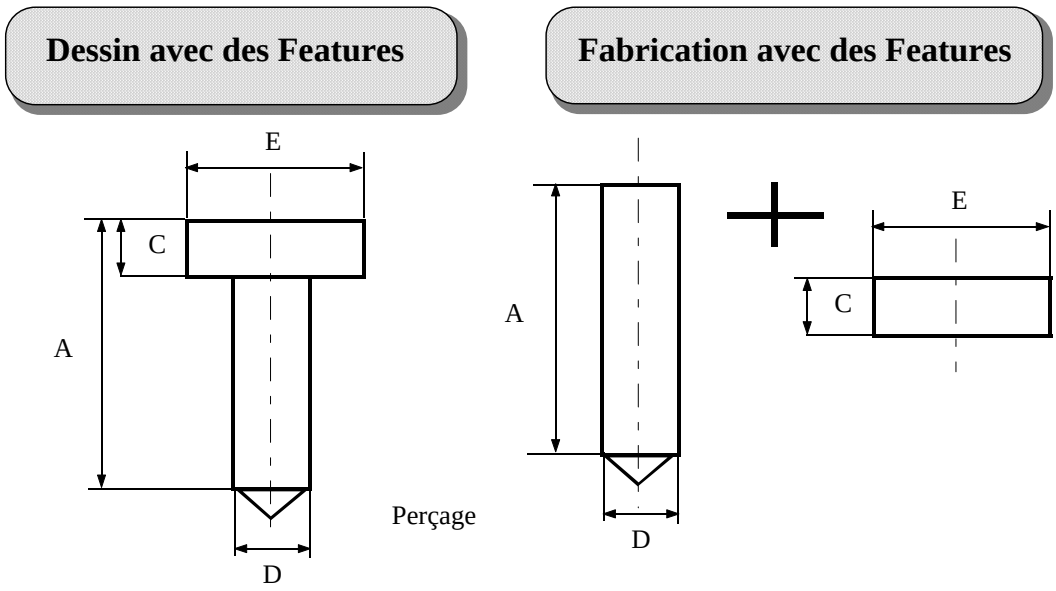


Figure 4 : Features de dessin et de fabrication

Pratt et Wilson ont apporté les fondements que nécessitent ces fonctionnalités d'une modélisation de " Feature " de formes dans le contexte d'un modeleur solide. Ces premiers travaux ont permis de catégoriser les différents types de " Feature ", explicites et implicites, sur la base de caractéristiques évaluables.

2.4. LANGAGES DE DESCRIPTIONS DE " FEATURES " [KRAUSE]

Les " Features " sont principalement des objets contenant de l'information. Aussi, un ensemble de données n'apporte de la connaissance au récepteur que si la sémantique, et la structure (organisation, relations entre ces données) de ces données sont connues. La détermination de ces structures et de cette sémantique constitue une des tâches principales de la modélisation des systèmes d'informations. A partir de données, pour lesquelles la sémantique et la structure ont été préalablement définies, il est alors possible de déduire un ensemble d'informations.

Pour réaliser la plus grande flexibilité possible tout en respectant les spécifications relatives aux applications, et en même temps l'indépendance du système, des langages existent pour la description des " Features " génériques.

Le problème soulevé, par rapport au niveau de sémantique, est de trouver un équilibre entre des données neutres (à faible sémantique) et donc partageables et les données à sémantique forte mais dont la spécificité par rapport à une application les rend peu partageables. Ce problème d'équilibre est donc lié à la notion de langage commun.

Le langage P.D.G.L. (Part Design Graphic Language), conforme à la norme I.S.O. 10303, permet de faire l'interface entre le modeleur et utilisateur, rendant ainsi les informations indépendantes du logiciel. Ce langage permet d'améliorer la pérennité et portabilité de l'information.

2.5. LES BASES DE DONNÉES [MONY]

Actuellement, il n'existe pas de système de C.A.O. complètement intégré dans une base de données du commerce. Certains systèmes utilisent malgré tout des

bases de données relationnelles pour faire leur gestion de documents, mais la description des modèles C.A.O. se fait par fichiers.

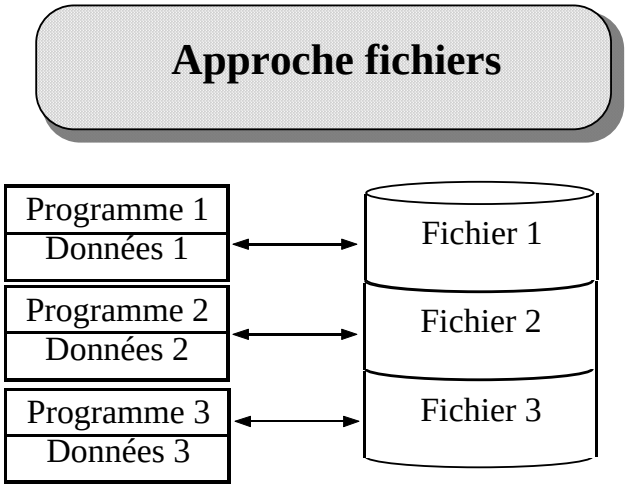


Figure 5 : Approche par fichiers

Les avantages d'une description en base de données d'un modèle de C.A.O. sont les suivants :

- le travail sur fichiers impose qu'ils soient stockés en mémoire centrale qui est limitée. Or les S.G.B.D.[6] gèrent non seulement la mémoire centrale mais aussi les accès disques ;
- actuellement, il n'est pas possible d'avoir un accès multi-utilisateurs sur un modèle sans faire un découpage préalable (nuisance pour les problèmes de cohérence) ;
- la tendance actuelle est une C.A.O. sur station de travail, la gestion de données par fichiers ne permet pas d'avoir une gestion réécrite des données ;

[6] S.G.B.D. : Système de Gestion de Base de Données.

- la gestion des modifications, des relations ou contraintes entre pièces, l'utilisation de bibliothèque d'éléments standards se révèle complexe;
- l'évolution des modèles C.A.O., aujourd'hui géométriques, vers les modèles technologiques (modèles de produits) amènera à gérer des modèles beaucoup plus complexes (au niveau des structures) avec des données aussi bien graphiques qu'alphanumériques. De plus ces modèles devront pouvoir évoluer facilement pour être enrichis tout au long du cycle de la production.

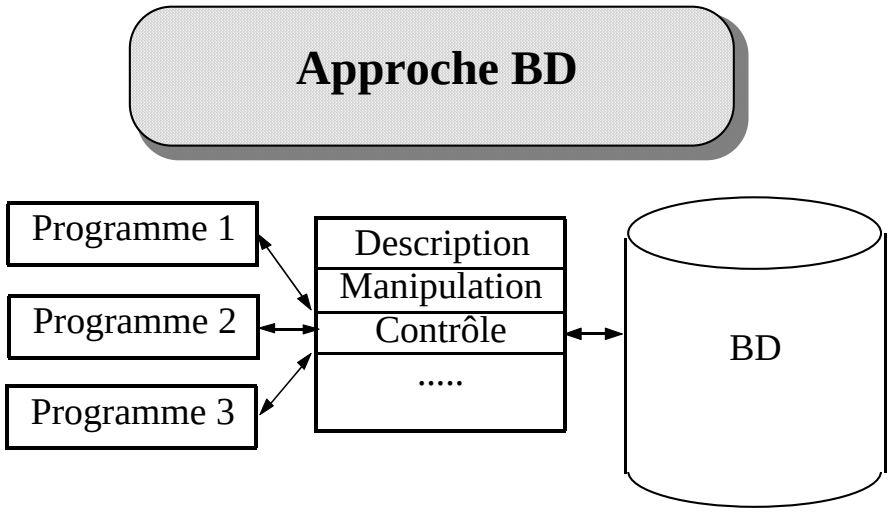


Figure 6 : Approche par base de données.

Dans les systèmes C.A.D., il est nécessaire de gérer deux types d'informations :

- les données techniques ;
- les données projet ou données produit.

4.2.6. Base de données techniques

Elle contient l'acquis technique de l'entreprise. Elle regroupe des données techniques indépendantes d'un projet précis et fournit des informations sur tout le cycle de production allant de la conception du produit, sa fabrication, sa livraison, sa maintenance et son recyclage.

On y trouve notamment les catalogues constructeurs, les archives d'études antérieures. Cette base est surtout utilisée en consultation, elle à pour objet essentiel : la fonction de documentation.

4.2.7. Base de données projet

C'est le noyau du système de C.A.O. Elle contient les données associées au projet proprement dit (Spécification, déroulement du projet, version et évolution de conception, spécification pour la fabrication, etc.) sur laquelle l'équipe de projet va travailler. Cette base est essentiellement dynamique, car les données doivent pouvoir être créées, consultées, modifiées, supprimées et ceci aussi bien au niveau des structures que des données elles-mêmes.

4.2.8. Les S.G.B.D. orientés objets

Les S.G.B.D. orientés objets ont une structure directement dédiée aux modélisations d'objet complexes. Ils offrent les avantages suivants :

- manipulation dynamique au même niveau du schéma et des instances ;
- système à identité forte (tous les objets et classes ont un nom) ce qui permet de modéliser fortement la sémantique des objets ;
- possibilité de représenter l'aspect dynamique des objets (méthodes) ce qui permet des contrôles d'intégrité et de cohérence puissants, et des

capacités déductives par un couplage avec un langage d'intelligence artificielle ;

- la possibilité d'associer un savoir faire à des classes pré-définies (bibliothèques), ce savoir étant hérité lors de l'instanciation (création de bibliothèques d'entités type) ;

- supporte la notion de version.

D'autres propriétés intéressantes sont en cours de développement, telles que :

- la gestion de l'incohérence ;

- le mécanisme des vues ;

- les problèmes de répartition sur station de travail.

5. LES SYSTÈMES CAD / CAM [DE MARTINO]

D'importants développements ont été réalisé dans le domaine de la conception, de la planification C.A.P.P., de la fabrication CAM, mais ces améliorations apportent des solutions pour des problèmes locaux. Un système C.A.P.P. peut générer automatiquement un plan de production mais seulement quand les utilisateurs fournissent une nouvelle description des fonctions de fabrication des pièces dessinées. Les mêmes choses arrivent pour les systèmes CAM. Dans ce cas, le système peut au moins utiliser les mêmes éléments géométriques résultant de la conception de la pièce, mais l'utilisateur doit fournir toutes les sorties du système C.A.P.P..

Le problème principal des systèmes de CAM / C.A.P.P. est qu'ils doivent être capables de comprendre la géométrie d'un produit à partir des données fournies par la C.A.O.. Lorsque l'on représente le perçage d'une pièce cubique, en projection plane, on dessine un carré et un cercle. Un système de F.A.O. reconnaît deux entités géométriques distinctes, alors qu'un dessinateur reconnaîtra la géométrie de la pièce. Les programmes actuels de recherche examinent d'autres approches, dans l'espoir de surmonter ces limitations [Schulz].

La solution globale pour résoudre ce problème est seulement possible à travers des " Features " de fabrication basées sur la représentation de pièce qui peuvent être comprises par tous ces systèmes.

Pour obtenir une telle représentation, il est possible d'utiliser trois méthodologies [De Jong]:

- L'identification de "Features" : L'utilisateur sélectionne de façon interactive à l'écran, les éléments du modèle géométrique déjà réalisé et définit des " Features " ;
- Reconnaissance de " Features " : Les modèles géométriques sont déjà définis conventionnellement et c'est un programme informatique qui identifie automatiquement les " Features " ;
- La conception de " Features " : Le modèle de pièce est directement construit avec des " Features " déjà définies dans le système.

Ces trois approches évaluent la nécessité d'attacher des informations implicites pour la représentation de constituants, et regardent la pièce comme un certain nombre de *Features* spécifiques qui ont pour effet la connaissance de la fonction de la pièce et la manière avec laquelle elle est réalisée. Elles construisent toutes les trois une représentation de la définition de la pièce en termes de " Features ". Cependant, les méthodes pour obtenir ces informations sont complètement différentes.

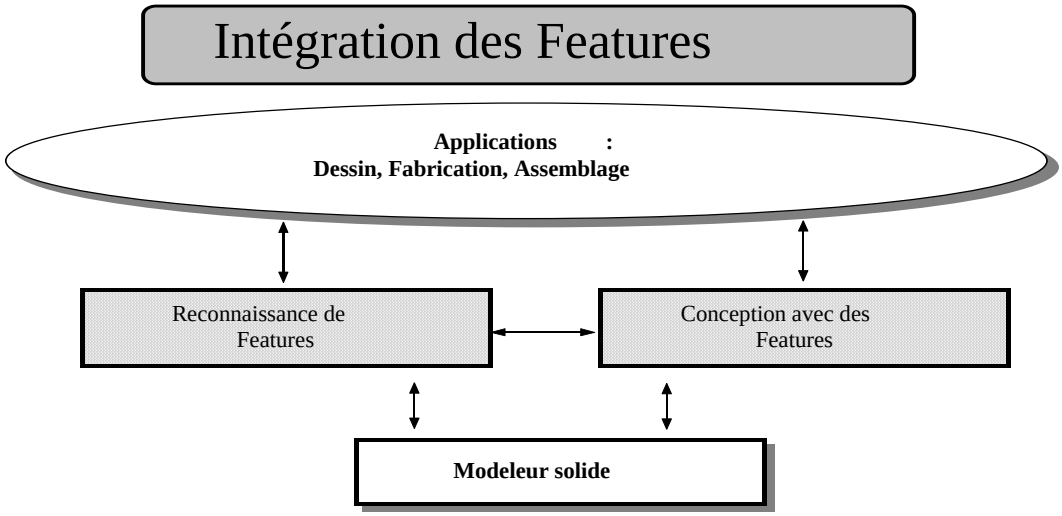


Figure 7 : Intégration des Features.

5.1. RECONNAISSANCE DE " FEATURES " [FIELDS] [PEROTTI]

Dans l'approche de reconnaissance de " Features ", le dessinateur utilise tout d'abord un modeleur solide conventionnel pour créer le modèle de sa pièce. Le modèle est alors soumis à l'opération d'un post processeur qui essaye de décomposer la géométrie en " Features ". Pour permettre cela, l'ordinateur dispose d'une librairie de " Features " qu'il comprend et qui peuvent être fabriquées. La représentation de la pièce peut alors être construite en termes de " Features ". Une telle base de " Features " peut, en théorie, être soumise à un système à base de règles de connaissance pour l'analyser.

Le problème de cette approche est que le post processeur doit être capable de reconnaître une " Feature ". La liberté que donne à l'utilisateur les modeleurs actuels fournit de nouvelles méthodes pour la construction d'un modèle ; on peut donc penser que le dessinateur sera capable d'inclure des " Features " qui ne soient pas reconnaissables.

5.2. CONCEPTION AVEC DES " FEATURES " [HOUTEN]

Dans l'approche de conception avec des " Features ", le modèle est construit en utilisant un ensemble de " Features " spécifiques, sélectionnées par le dessinateur. Dans le même temps, l'utilisateur peut être invité à introduire des informations supplémentaires sur celles-ci que le système connaît et qui sont nécessaires pour l'analyse qui suit.

Aussi pendant que le dessinateur utilise le système, une base de " Features " est construite en parallèle avec les fichiers d'informations du modeleur, et celui-ci devient juste un moyen pour représenter la géométrie de la pièce à l'écran. Le principal avantage de cette méthode est, que le modèle est construit à partir de " Features " que le système comprend, et qu'il n'est pas possible au le dessinateur de construire une " Feature " que l'ordinateur ne peut pas globalement comprendre [Frans].

5.3. CONTRAINTES DE FABRICATION ET " FEATURES " [DELBRESSINE]

Le but des " Features " orientées vers la fabrication est de transposer des " Features " spécifiques dans un produit pour qu'elles puissent remplir les fonctions souhaitées. La conception d'un produit se fait en deux phases : une phase conceptuelle et une phase géométrique. La phase géométrique transforme les idées dans le dessin, alors que la phase conceptuelle s'occupe de formaliser ces idées pour la production. La phase conceptuelle est maintenant incluse dans le processus de conception, sans être cependant formalisée.

Une première proposition pour améliorer le dessin est la représentation complète et sans ambiguïté du produit de sorte qu'il puisse être fabriqué et accomplir

la fonction désirée. Cette solution nécessite la définition de restrictions de fabrication à l'étape de la conception géométrique. Il y a deux façons de définir les restrictions dues à la fabrication :

- vérifier complètement le dessin ;
- vérifier l'état initial de chaque " Feature " et toutes les transformations qui leur sont appliquées.

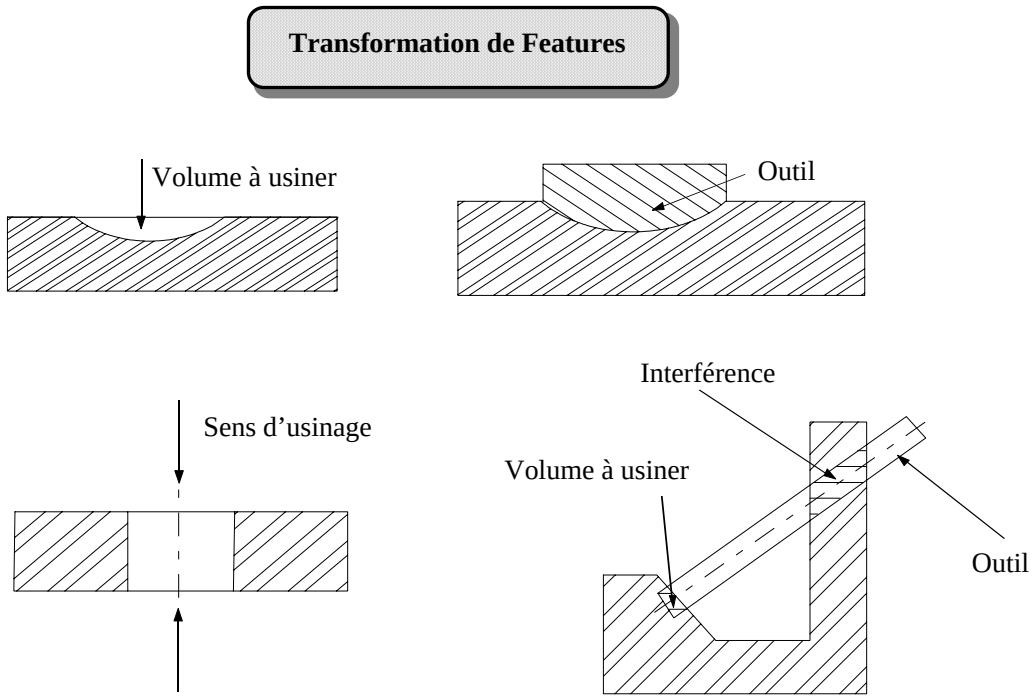


Figure 8 : Transformations de " Features ".

La première approche est actuellement trop complexe pour être performante automatiquement. Pour utiliser la seconde approche, il est nécessaire de définir des transformations de " Features ". Une " Feature " fabriquable doit comprendre une combinaison de un ou plusieurs outils, machines et configurations des phases de fabrication. Les " Features " comprennent deux formes géométriques : l'état géométrique initial et l'état géométrique final avec un ensemble de règles

d'application. Ces règles d'application assurent la prise en compte des contraintes de fabrication et assure la possibilité de fabrication du produit.

3.4. ARCHITECTURE DES SYSTÈMES CAD / CAM

L'architecture des système CAD / CAM est comparable à la structure du processus d'innovation.

Les premiers systèmes, à une structure séquentielle, n'étaient que des îlots d'optimisation. On utilise une succession de logiciels, C.A.O., F.A.O., G.P.A.O., qui apportent une aide à chacune des personnes intervenant dans son domaine. La communication entre ces systèmes s'effectuant à travers des interfaces. Ce type d'interface se fait soit, à l'aide de fichiers d'échange spécifique, soit à l'aide de fichiers neutres.

La seconde génération de système utilise le principe de l'ingénierie concourante, et prône l'intégration dans une application unique de ces trois domaines. Ce type de structure est utilisé dans les modeleurs orientés objets (P.T.C.) ou dans les modeleurs hybrides (C.A.D.D.S.).

6. INTÉGRATION DE " FEATURES " DANS LES SYSTÈMES CAD / CAM

De nombreux systèmes actuels intègrent la notion de " Features ". Certains sont encore au stade de prototype, d'autres sont déjà commercialisés. Ces applications se séparent en deux groupes :

- dans le premier, on trouve des modeleurs volumiques traditionnels auxquels on a rajouté une interface utilisateur qui gère conjointement une base de données de " Features " (orientée conception, fabrication, etc.) et le modeleur géométrique ;
- dans le second, on trouve des modeleurs géométriques orientés objets. Dans ce cas, il est aisé pour le modeleur de manipuler la notion de " Features ".Et ainsi, chaque objet géométrique peut contenir, en plus des informations de type topologique, des informations technologiques.

6.1. LES MODELEURS HYBRIDES

Dans un système CAD hybride, le dessinateur utilise une méthodologie de conception basée sur des " Features ". Lorsque l'on dessine une pièce ou que l'on définit des attributs technologiques, le dessinateur intervient avec la base d'informations des formes " Features ". En considérant cela, le système peut non seulement supporter le processus de conception, mais aussi vérifier la faisabilité de la pièce et fournir des représentations de pièces qui peuvent être utilisées par les systèmes C.A.P.P. et CAM.

Le module de dessin apporte toutes les fonctions nécessaires pour la

validation et la manipulation des " Features ", l'identification automatique des interactions entre les " Features " et la définition des attributs technologiques tels que les tolérances et les états de surfaces. Il gère les fonctions pour maintenir la cohérence du modèle de la pièce et les informations géométriques. Pour cela, le système utilise une librairie décrivant des " Features ", des modèles de production, des connaissances sur les tolérances et la fabrication des " Features ". Pour la représentation géométrique de la pièce et l'interface graphique avec l'utilisateur, le système communique avec le modeleur géométrique. [Schulz]

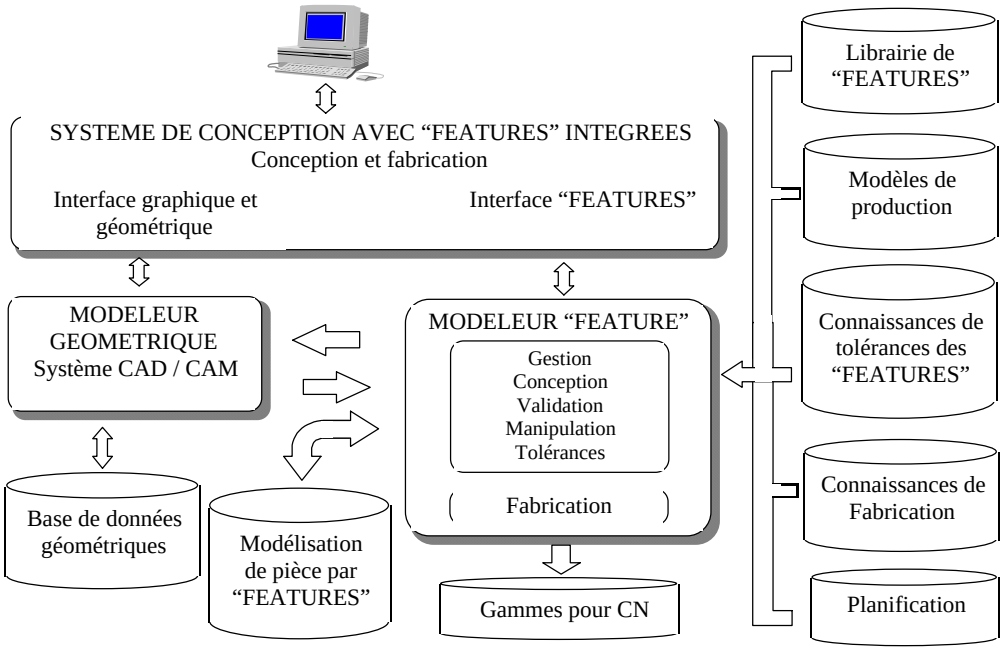


Figure 9 : Modeleur hybride

6.2. LES MODELEURS ORIENTÉS OBJETS

Les modeleurs orientés objets utilisent toutes les possibilités basées sur ce type de programmation, soit la notion d'héritage, de polymorphisme. Ils gèrent ainsi la réactualisation et la modification entre les objets et utilisent un fichier unique pour la description de chaque pièce, tant du point de vue de la conception, de l'assemblage, de la fabrication et de la mise en plan.

DEUXIÈME PARTIE :

RAPPORT DE RECHERCHE

7. INTRODUCTION

Le but de l'introduction de la fabrication dès la conception est de tenir compte des moyens utilisés et des contraintes qui leurs sont associées pour la fabrication. Cette méthodologie a pour objet, la diminution des coûts de fabrication, et surtout de réduire les coûts dus à certaines décisions prises lors de la conception. Cependant, ces contraintes peuvent aussi conduire à une augmentation des coûts de production (de même que la T.G.A.O.[7] pour la G.P.A.O.[8]). Le but de cette étude est de trouver à quels stades de la conception le dessinateur doit tenir compte des contraintes dues à la fabrication de son projet, et comment les introduire dans un modeleur géométrique. Ceci afin de s'assurer que le modèle conçu sera réalisable par un procédé de fabrication prédéterminé.

Ce sujet rentre dans le cadre de l'ingénierie simultanée qui prône une meilleure intégration entre la C.A.O et la F.A.O.

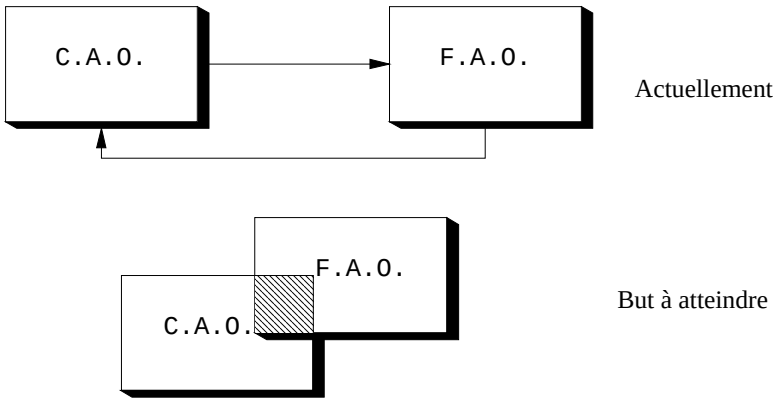


Figure 10 : Ingénierie simultanée

[7] T.G.A.O. :Technologie de Groupe Assisté par Ordinateur.

[8] G.P.A.O. : Gestion de Production Assisté par Ordinateur.

8. DÉMARCHE DE CONCEPTION

Pour connaître les contraintes de fabrication à apporter dans un modeleur géométrique, nous allons étudier un système de transmission de puissance, constitué d'un réducteur de vitesse.

8.1. DÉROULEMENT DE LA CONCEPTION

La création d'un réducteur de vitesse tient compte d'un certain nombre de contraintes. Celles-ci peuvent être de type encombrement, prix, durée de vie, bruit, poids, etc..

Pour créer ce réducteur de vitesse, il est nécessaire de connaître à quel moment de la conception, le dessinateur devra en tenir compte. Ces contraintes peuvent être de plusieurs types, soit par exemple : contraintes de taillages pour les engrenages (interférences), contraintes de montages pour les clavettes (suivant la forme de celle-ci), contraintes de fabrication des arbres cannelés (rayons de raccordement nuls), contraintes dimensionnelles (épaisseur minimale du carter pour la fonderie).

Pour obtenir ces résultats, il est nécessaire de concevoir complètement le mécanisme, afin de connaître à chaque étape de la conception, les décisions à prendre par rapport aux contraintes de fabrication.

8.1.1. Dimensions

La principale contrainte rentrant dans la conception d'un réducteur est le rapport de réduction. Pour ce faire, il est nécessaire de choisir un schéma cinématique permettant d'atteindre ce but. Dans un premier temps, l'ingénieur doit déterminer les rapports de réduction de chaque train d'engrenages ainsi que leurs

caractéristiques (type de denture, module, nombre de dents, etc.) et, à partir des efforts à transmettre et les contraintes appliquées aux dentures, la matière à utiliser. Dans les rapports de réduction, le nombre minimum de dents est soumis à la méthode de fabrication. Ces contraintes portent sur le mode de taillage des dentures, et entraînent le choix d'un usinage particulier (taillage par fraise au module ou crémaillère).

D'autres difficultés peuvent apparaître dans le type d'implantation de certains composants, tels que les clavettes, les goupilles, les roulements, etc.. Ces informations nous permettent de définir les surfaces fonctionnelles de notre pièce.

8.1.2. Fabrication

Le choix d'une méthode de fabrication n'est pas uniquement déterminé à partir des contraintes fonctionnelles des pièces. Un type de processus de fabrication peut être choisi en fonction du prix de la pièce, de la taille de la production (unitaire, petite série, moyenne série ou grande série). Toutes ces exigences peuvent obliger, dans le cas de la fabrication d'un carter, à choisir celui fabriqué soit en mécano soudé soit par fonderie (fonderie en coquille, au sable ou cire perdue) soit par usinage.

Ces contraintes agissent sur les surfaces non fonctionnelles de la pièce mais non sur les surfaces fonctionnelles. Pour concevoir les surfaces non fonctionnelles, il est donc nécessaire de connaître la méthode d'obtention de la pièce. A partir de ces connaissances, on peut introduire un certain nombre de contraintes inhérentes à ce type de fabrication.

8.1.3. Usinage

Si la fabrication par fraisage, par perçage, par brochage, par rectification ou par tournage, est choisie, il est possible de concevoir en fonction des règles

qu'impose ce type de fabrication. Pour réaliser une pièce, il est donc nécessaire de faire une distinction entre les surfaces fonctionnelles et les surfaces non fonctionnelles. Les premières sont surtout contraintes par la conception, et les autres, étant moins importantes, sont liées aux moyens de production.

8.2. BUTS À ATTEINDRE

Dans le cadre de ce sujet de recherche, nous avons choisi de nous intéresser à la fabrication de poches. Nous ne chercherons pas à connaître si la surface est fonctionnelle ou ne l'est pas, et nous étudierons des poches répondant à quelques exigences de forme :

- elles ne sont pas débouchantes .
- il n'y a pas d'intersection avec d'autres poches.

Mais elles peuvent contenir :

- des îlots ;
- des perçages ou des évidements de matières ;

Pour mettre en place ces contraintes de fabrication, nous avons choisi le modeleur géométrique Pro-Engineer de Parametric Technology Corporation qui supporte la notion de « Feature » et permet, à partir de son interface de programmation Pro-Develop, d'incorporer de nouvelles fonctions.

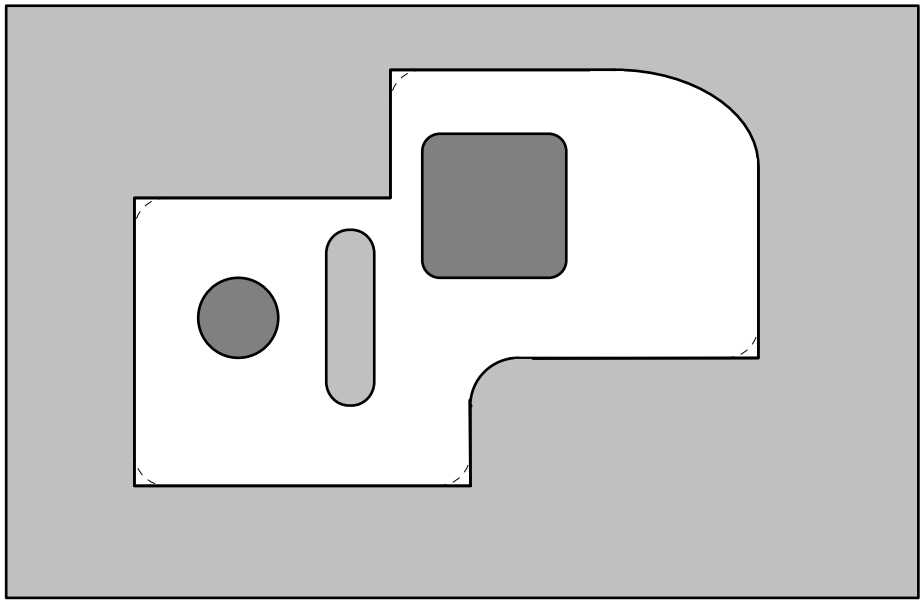


Figure 11 : Type de poches à vérifier

Notre application a pour objet, de montrer comment une poche peut être usinée par fraisage et avec quel outil. Pour permettre d'apporter ces résultats, nous devons être capables de construire un parcours d'usinage, qui correspond au cycle d'ébauche du contour de la poche. Ce résultat permet d'aider au choix de l'outil adéquat.

9. PRO-ENGINEER ET PRO-DEVELOP

9.1. INTRODUCTION DU MODELEUR PRO-ENGINEER

Le modeleur Pro-Engineer est un modeleur solide dont le noyau est basé sur une modélisation par limites plus connue sous le nom de Boundary Solid Representation (B-Rep). De plus, Pro-Engineer dispose de fonctions de génération de solides basées sur la notion d'entités fonctionnelles (modélisation par Features) paramétrables. Ces Features peuvent correspondre à des notions de fabrication (perçages, poches, nervures, arbres, congés, chanfreins, etc.).

9.2. INTRODUCTION DE L'INTERFACE DE PROGRAMMATION DE PRO-DEVELOP

Pro-Develop est l'interface de programmation du modeleur géométrique Pro-Engineer. Cette interface permet aux créateurs de logiciels de personnaliser les fonctions de Pro-Engineer, en autorisant la création de nouvelles applications pouvant être directement et complètement intégrées dans l'environnement du modeleur.

Pro-Develop consiste en une bibliothèque de fonctions de bases écrites en langage « C » qui permet d'accéder partiellement à l'interface utilisateur de Pro-Engineer ainsi qu'à la base de données du modeleur.

Les accès aux modules de Pro-Engineer, tel que Pro-Manufacturing, ne sont pas possibles.

9.3. ACCÈS À LA BASE DE DONNÉES

Pro-Develop offre un certain nombre de fonctions pour exploiter la base de données géométriques. Ces fonctions permettent de décrire la géométrie des objets sous leur représentation B-Rep et sous forme de Features. Pour décrire l'objet, la géométrie se décompose en trois niveaux : les surfaces, les contours, les arêtes. Ces informations représentées, en langage informatique, sous forme de pointeurs chaînés décrivent, de la même façon, la géométrie B-Rep et les Features.

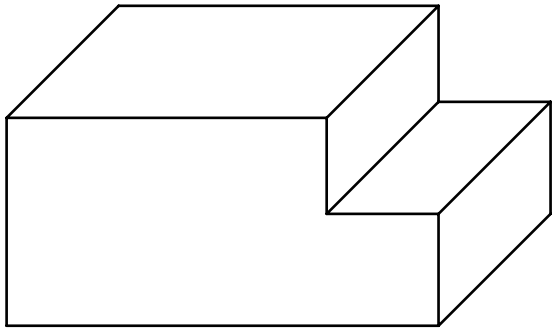


Figure 12 : Représentation d'un objet.

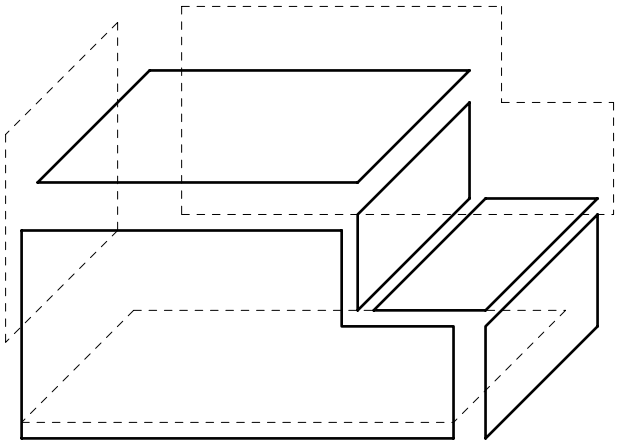


Figure 13 : Décomposition de la géométrie d'un objet, les surfaces.

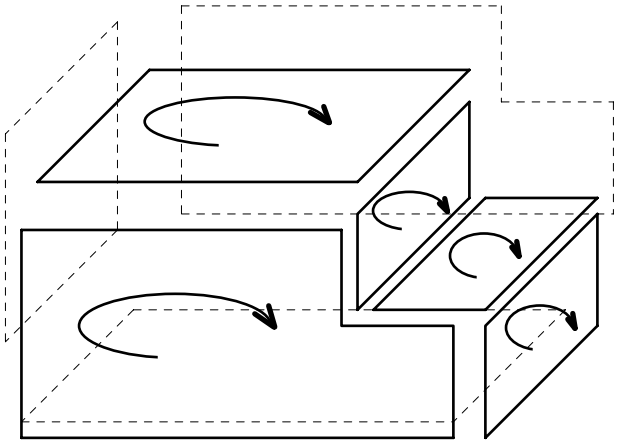


Figure 14 : Décomposition de la géométrie d'un objet, les entités.

Dans notre étude de Pro-Develop, nous n'avons pas réussi à utiliser correctement les informations contenues dans le Sketch 2D. Actuellement, aucune fonction de Pro-Develop ne permet d'accéder à la structure de données de l'esquisse et la fonction qui permet l'enregistrement des données sur fichiers (.sec) n'est pas efficace car elle ne donne aucune information concernant son placement sur la pièce. Pour la suite de notre recherche, nous avons choisi d'utiliser uniquement la géométrie B-Rep de la pièce.

9.4. LES MENUS

Nous pouvons rajouter, dans les menus de Pro-Engineer, des commandes spécifiques à notre application. Les commandes introduites dans les menus permettent soit de personnaliser des fonctions existantes, soit d'apporter de nouvelles fonctions. Cependant, il n'est pas possible d'utiliser une commande de Pro-Engineer, si elle n'existe pas dans la librairie de Pro-Develop. Cet inconvénient est dû aux limitations accès aux fonctions de Pro-Engineer par Pro-Develop.

Un extrait du manuel de Pro-Engineer, traduit en français, est présent en Annexes (§ 1.2. Les menus).

10. L'INTERFACE X11

Pour permettre de développer une interface graphique sous Silicon Graphic, nous avons étudié l'interface X11. Cette librairie, écrite en langage « C » est disponible sur toutes les stations de travail UNIX. Cette condition nous a permis de créer un programme simple et efficace.

Nous avons regroupé en Annexes (§ 2. X11) de nombreuses informations concernant les avantages de X11 et des exemples d'application.

11. APPLICATION

11.1. INTRODUCTION

L'introduction de contraintes de fabrication dans Pro-Engineer pose plusieurs problèmes. Pour être optimum, ces contraintes doivent apparaître le plus tôt possible dans la démarche de conception. Dans Pro-Engineer, la création d'une poche se déroule en quatre temps :

- esquisse de la section de la poche, dessin à main levée du contour de la poche ;
- cotation du contour ;
- validation ou régénération de la fonction ;
- introduction de la profondeur de la poche.

La vérification ou modification d'une *Feature poche*, dans le contexte *Pro-Engineer*, peut donc se réaliser sur deux niveaux :

- au niveau du *Sketch* ;
- après validation de la fonction.

Aucune modification des dimensions n'est viable tant que le Sketch n'a pas été complètement validé. De plus, le Sketch ne donne aucune information sur l'emplacement de la poche. En effet, celui-ci peut avoir des intersections avec d'autres *Features*. La forme définitive de la poche n'est visible qu'après la validation du Sketch et l'introduction par l'utilisateur, de la profondeur de celle-ci.

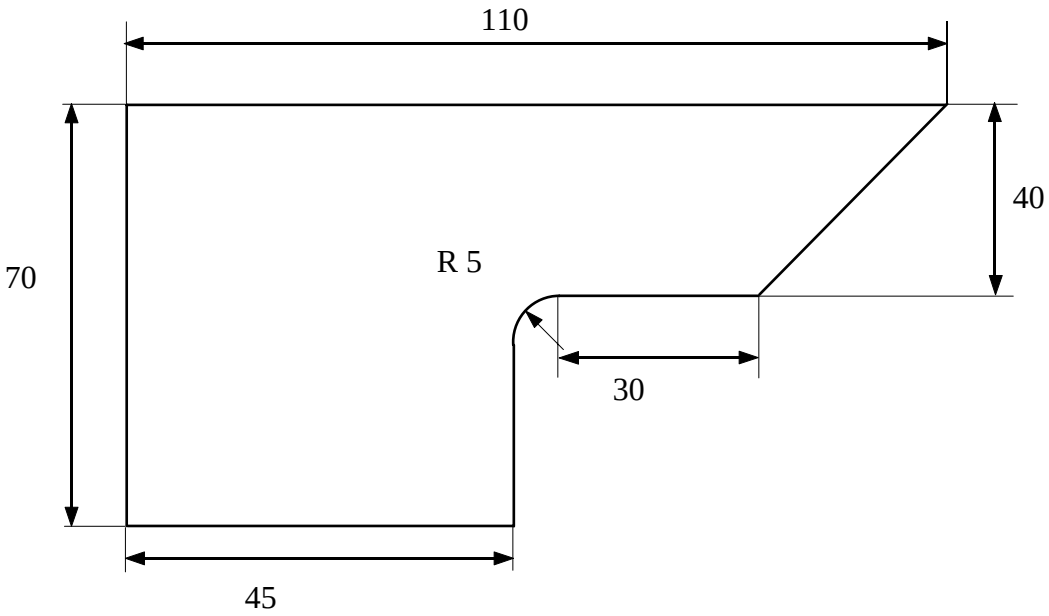


Figure 15 : Exemple de Sketch de Pro-Engineer.

Ces constatations nous conduisent à introduire nos contraintes de fabrication après avoir validé la Feature poche.

Après avoir choisi à quel niveau agit notre application, nous devons étudier la méthode à utiliser pour introduire des contraintes de fabrication. La différence entre proposer et imposer des solutions est importante, car elle peut s'appliquer différemment sur le logiciel.

- Proposer des solutions peut se faire en affichant une fenêtre indépendante dans laquelle est dessinée la fonction et en ajoutant, d'une autre couleur, les modifications à apporter.
- Par contre, imposer des solutions ne peut se faire qu'en modifiant le *Sketch* dessiné dans *Pro-Engineer*.

Si ces propositions sont difficilement réalisables dans le module *Sketch* de *Pro-Engineer*, il est possible de suggérer à l'utilisateur un certain nombre de fonctions lui permettant de compléter la réalisation de la poche.

11.2. STRUCTURE DE L'APPLICATION

Pour permettre une meilleure utilisation de l'application, deux programmes ont été créés. Le premier fonctionne simultanément avec *Pro-Engineer*. Il permet la lecture de la base de données géométriques, lorsque l'utilisateur demande une vérification de la faisabilité des *Features poches* incorporées dans le dessin. Il crée deux fichiers d'échange nécessaires au fonctionnement du second programme pour visualiser le profil de la poche. Le premier, *piece.dat*, contient la géométrie B-Rep de la pièce et le second, *feature.dat*, la représentation B-Rep des *Features poches.* Cette structure permet, à l'utilisateur, d'utiliser en même temps les commandes de *Pro-Engineer* et celles de notre application. Il est ainsi possible de réaliser des rotations, des visualisations réelles ou des agrandissements de la pièce pour examiner toutes les poches et d'exécuter les opérations nécessaires à la réalisation de celles-ci dans le cadre d'opérations de fraisage.

11.3. FILTRE D'ENTRÉE ENTRE PRO-ENGINEER ET NOTRE APPLICATION

Dans le cadre du sujet, nous avons choisi d'étudier la fonction poche réalisée par fraisage. Cependant, dans le contexte de *Pro-Engineer*, la *Feature* poche peut prendre de nombreuses formes.

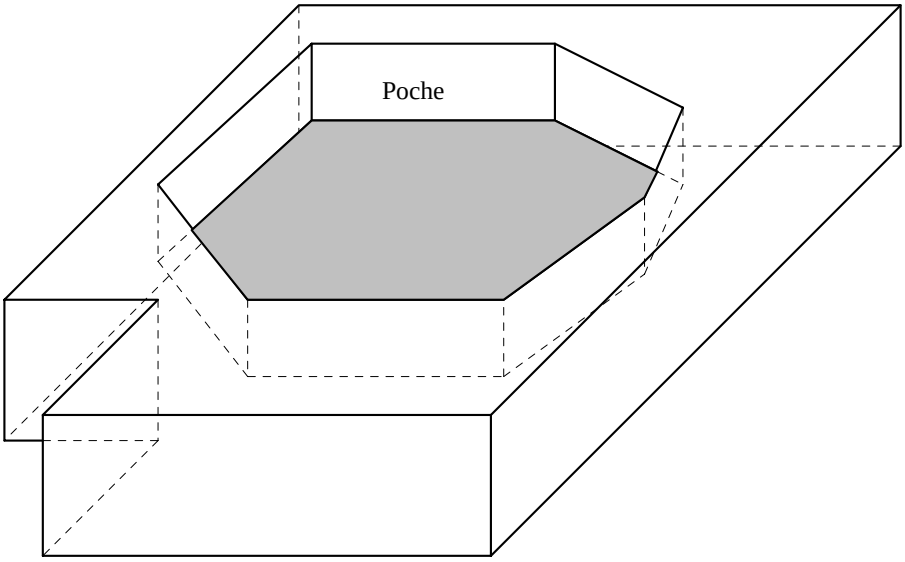


Figure 16 : Fonction poche, Extrusion, Borgne.

La *Feature poche* permet de réaliser des poches de type extrusion, révolution, balayage, lissage, avancée.

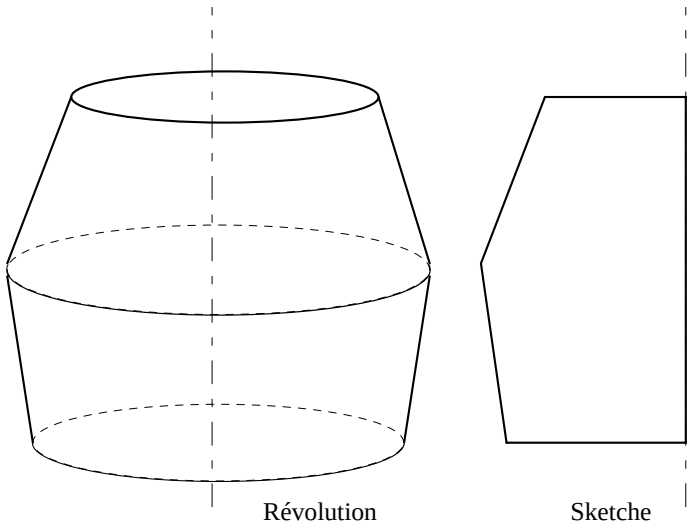


Figure 17 : Fonction poche, Révolution, 360°.

Pour n'utiliser que les « vraies » fonctions poches, il est nécessaire de filtrer les entrées. Pour ce faire, nous avons utilisé les fonctions *prodb_get_feat_ids* et *prodb_get_feat_type*. Celles-ci permettent de connaître la liste des *Features* utilisées dans un objet et, pour chacune d'entre elles, leurs attributs. (Voir Annexes § 1.1. Les Features)

11.4. UTILISATION DES INFORMATIONS GÉOMÉTRIQUES

Pour permettre une meilleure compréhension de la Feature poche, nous utilisons simultanément des informations de type B-Rep et de type Feature. Ces informations sont complémentaires et permettent de mieux connaître le placement de la poche. Dans le cadre de ce sujet, nous avons décidé de n'utiliser que les entités de type ligne et arc. En effet, se servir de splines et de N.U.R.B.S. aurait nécessité le développement de nombreuses fonctions d'intersection (ligne-spline, arc-spline, etc.).

11.5. CRITÈRES D'IDENTIFICATIONS DE LA GÉOMÉTRIE

Les informations fournies par Pro-Engineer reposent sur la lecture de la structure B-Rep de l'objet. L'orientation des entités doit donc obéir à la règle de MOEBIUS (Voir Bibliographie § 2.2.1. Le modèle par limites). Pour étudier la forme de la poche, nous avons décidé de prendre comme élément de départ le contour formé par le fond de la poche. Celui-ci est comparable à celui dessiné dans le Sketch avec, en plus, sa forme définitive dans la pièce (placement, profondeur, interférences).

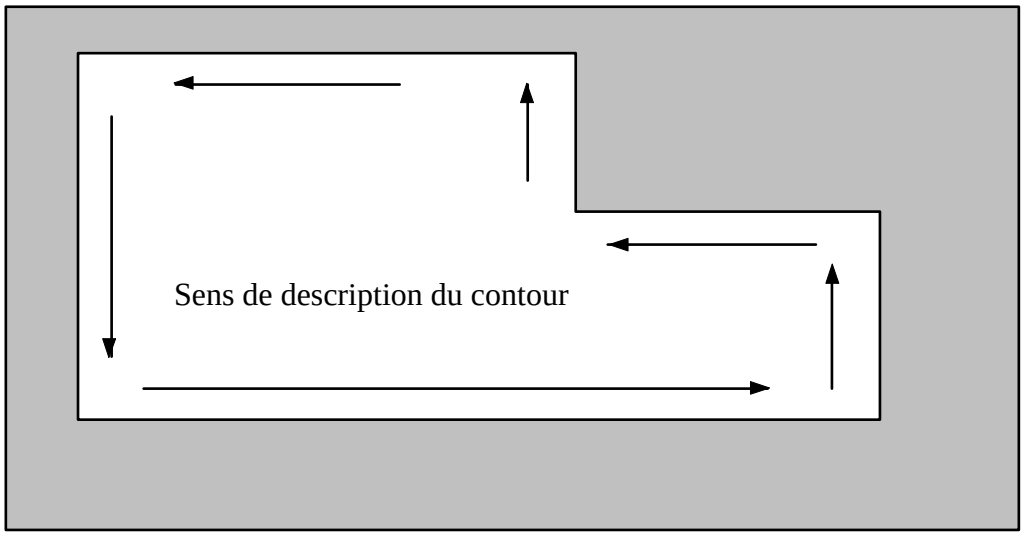


Figure 18 : Direction des arêtes d'un contour.

Les informations fournies par Pro-Engineer se présentent sous la forme de pointeurs chaînés. Chacune de ces entités représente, suivant son type, un segment, un arc, une spline ou une N.U.R.B.S.. Leur ordre indique la position de la matière et donc la position du parcours d'usinage par rapport au dessin de la poche. Pour une poche simple, comme celle dessinée ci-dessus, les informations données par la géométrie de la poche sont les suivantes :

- un contour pour la section de la poche ;
- des entités telles que des lignes ou des arcs pour définir les flans de la poche.

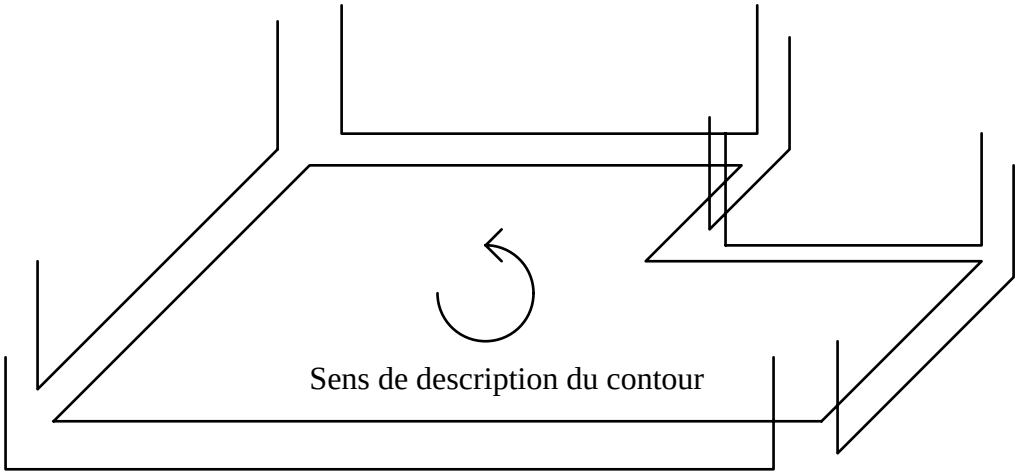


Figure 19 : Informations topologiques fournies par une Feature poche.

Lorsqu'une autre fonction vient s'ajouter à la fonction poche, le problème devient différent. En effet, si l'utilisateur dessine un congé ou un chanfrein sur l'une des arêtes de la poche, cela revient à « enlever » de la matière dans la poche.

Cette opération modifie le sens de description du contour de la poche de la manière suivante :

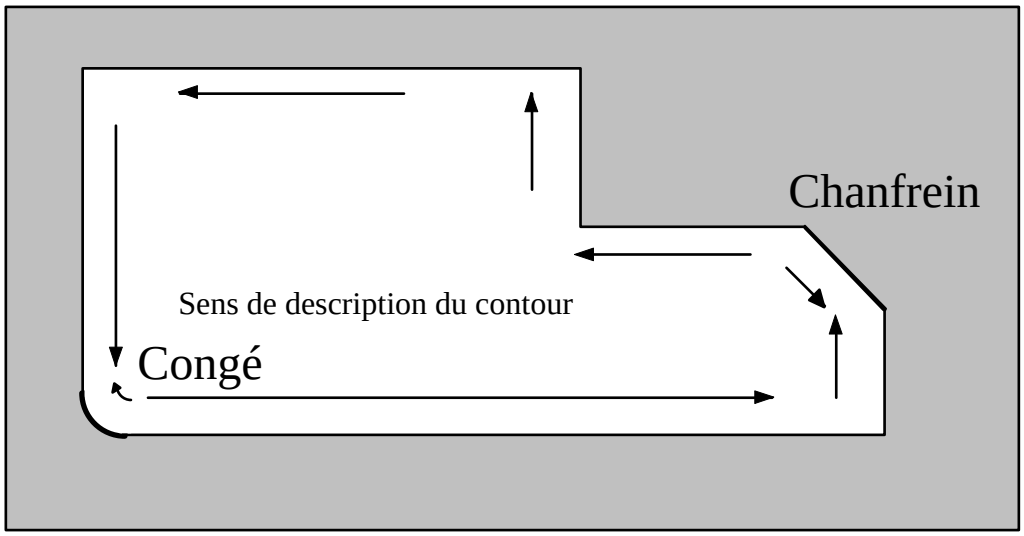


Figure 20 : Modification du sens de description d'un contour : un congé et chanfrein.

Il apparaît donc une inversion locale du sens de description du contour. Ce changement de sens pouvant se produire n'importe où sur le contour, il n'est pas possible de connaître avec certitude le segment qui n'obéit pas à la règle de MOEBIUS. La création d'un congé ou d'un chanfrein entraîne d'autres modifications pouvant aider à comprendre la géométrie de la poche. Dans la géométrie donnée par la Feature poche, les côtés appartenants aux congés et aux chanfreins ne sont pas présents.

Les différences de topologie sont montrées dans le dessin suivant :

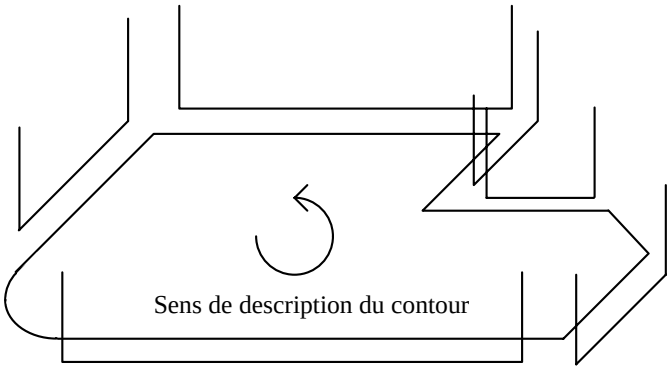


Figure 21 : Informations topologiques fournies par une Feature poche après création d'un congé et d'un chanfrein.

Ces informations pourraient suffire si ce phénomène n'arrivait pas lorsque la poche n'est pas complète.

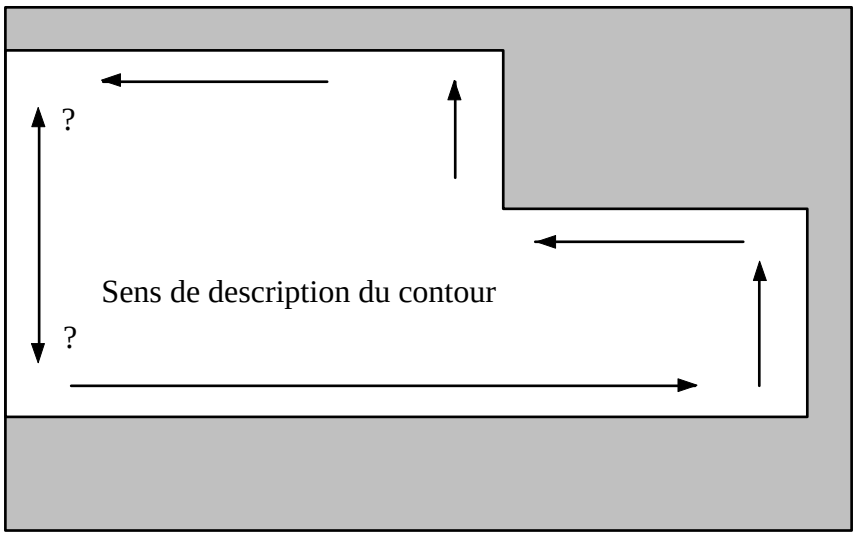


Figure 22 : Modification du placement d'une poche.

Lorsque la poche se situe en bordure de la pièce, la géométrie de la Feature est incomplète et l'orientation de l'entité (segment ou arc) située sur le bord extérieur est aléatoire.

Ce problème se retrouve dans la définition des îlots. La résolution de ce problème sera donnée dans le paragraphe suivant.

11.6. LES ÎLOTS

Sur une surface de la géométrie B-Rep, peuvent se trouver plusieurs contours. Cela peut être dû, soit à une discontinuité de la poche, soit à la présence d'un îlot.

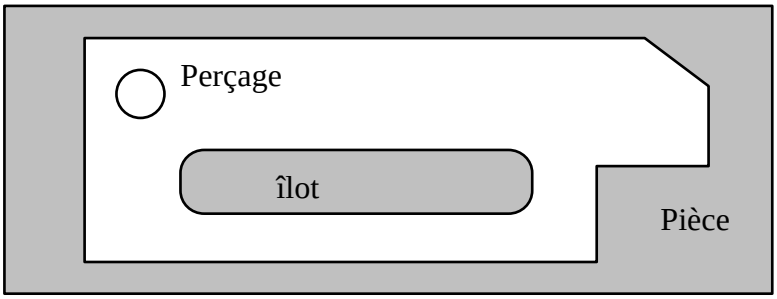
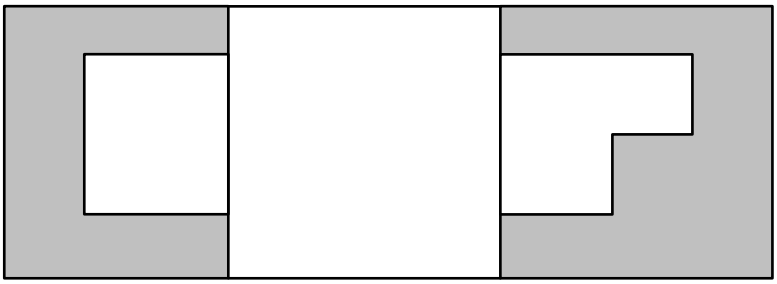


Figure 23 : Présence d'un îlot dans une poche.



Découpe ou placement incorrecte de la poche

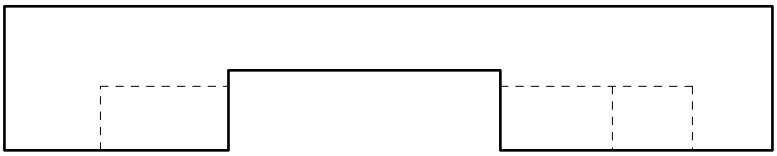


Figure 24 : Discontinuité dans le placement d'une poche.

Dans les deux cas, il faut pouvoir faire la différence entre le contour principal de la poche, et les contours annexes.

Dans un premier temps, nous allons chercher à connaître l'orientation de la surface d'appui de la poche. Dans ce but, nous devons calculer la normale au contour. Cette précision nous est donnée dans la géométrie B-Rep de la pièce. En effet, pour définir une surface, Pro-Engineer nous fournit trois vecteurs plus une origine. Nous devons donc parcourir toute la géométrie et reconnaître au minimum

deux entités identiques entre le contour de la poche et le contour présent sur la surface.

Grâce à ces informations, nous possédons une matrice de projection qui nous permet de transformer notre problème en un problème de géométrie plane.

Nous savons que, dans une poche, si une entité est présente deux fois dans la géométrie de la Feature poche, son orientation est correcte. Pour connaître l'orientation des îlots, nous devons nous servir d'informations données par la géométrie B-Rep.

Nous avons constaté que, quelque soit le sens de l'îlot (protrusion ou découpe), l'orientation du contour était identique.

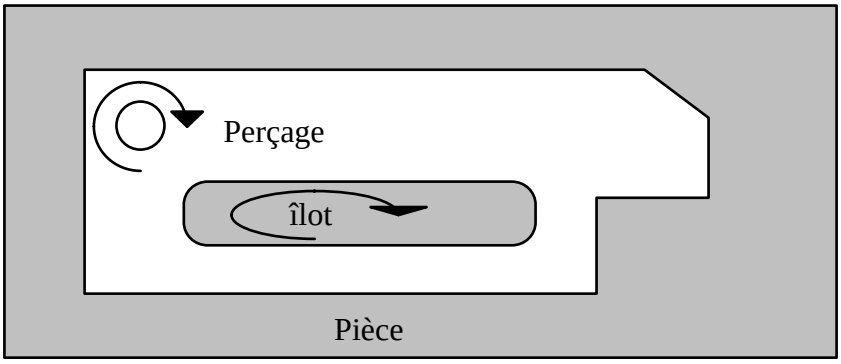


Figure 25 : Orientation des contours des îlots.

Dans le cas de notre étude, les contours dus à des opérations de perçage ou de découpe à l'intérieur de la poche ne nous intéressent pas dans le traitement de l'usinage de celle-ci. Pour trouver, l'orientation des autres entités nous avons fait appel à la géométrie B-Rep de la pièce. Nous cherchons à calculer un vecteur, dont l'origine coïncide avec un point du profil de la poche, puis nous effectuons notre test de placement à l'aide d'un produit scalaire.

Ces informations permettent de calculer, pour toutes entités d'un contour, son orientation.

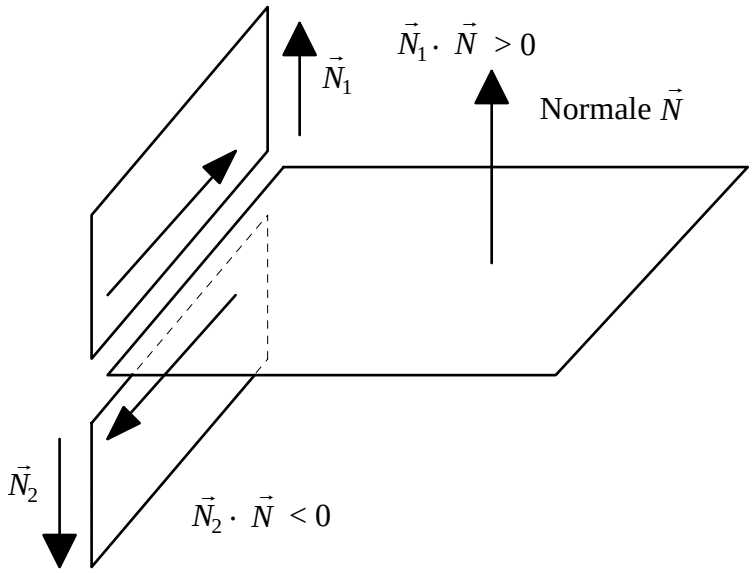


Figure 26 : Test de normale.

Ce test ne nous permet pas de résoudre tous les problèmes, car dans certaines situations, il est possible de mettre en défaut cette règle. Pour indication, nous allons montrer une situation où le résultat peut être mauvais.

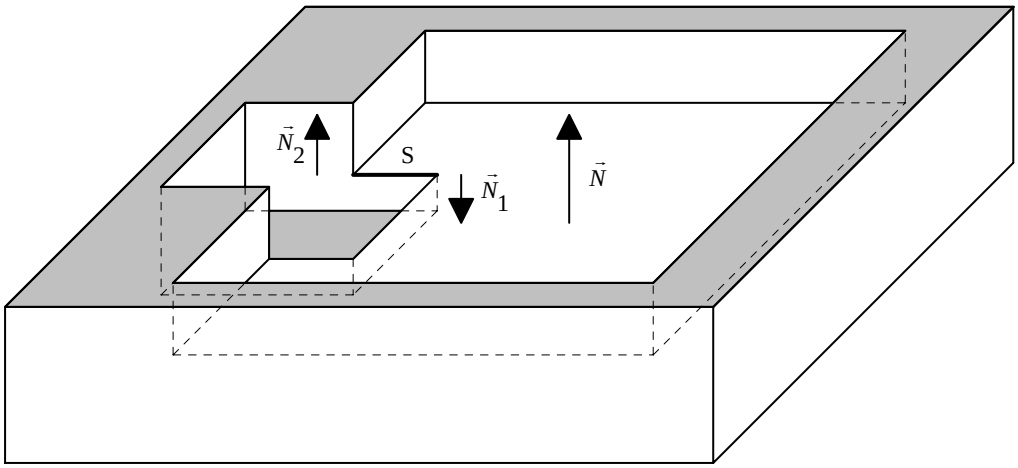


Figure 27 : Erreur du test de la normale.

Suivant l'orientation du segment S, le test de la normale se fait entre les vecteurs $\vec{N}_1$ et $\vec{N}$ ou entre les vecteurs $\vec{N}_2$ et $\vec{N}$. Ce résultat est fonction de la géométrie du flan de la poche.

Cette configuration particulière a été écartée car elle ne correspond pas au profil de poche défini dans nos conditions initiales (Voir § 2.2 But à atteindre).

11.7. PARCOURS D'USINAGE

Pour savoir si la poche peut être usinée, il est utile de définir un parcours d'usinage. Cette opération est réalisée en exploitant les résultats obtenus précédemment. Pour chaque entité constituant le contour, des règles sont à respecter.

Pour les segments, nous effectuons une translation de l'entité de la valeur d'un rayon de fraise.

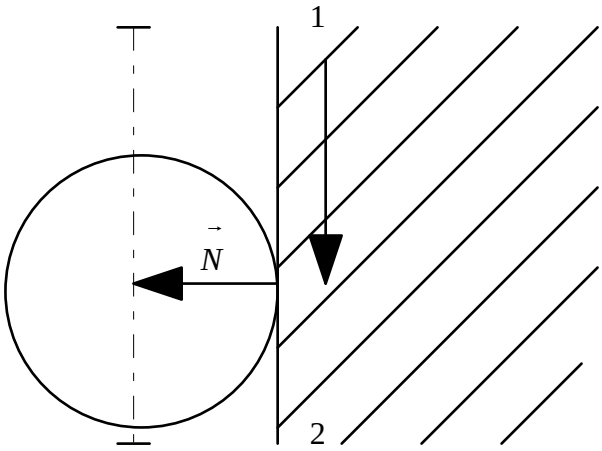


Figure 28 : Translation d'un segment.

Pour les arcs, l'orientation du parcours est donnée par sa normale.

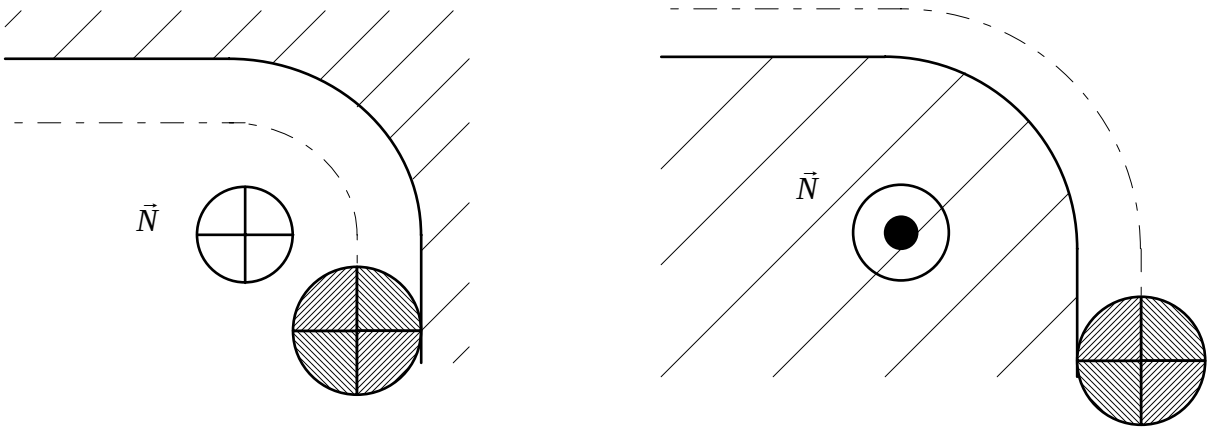


Figure 29 : Translation d'un arc.

Après avoir considéré chaque entité, nous devons effectuer les intersections entre chaque entité et adapter le parcours d'usinage aux règles de fabrication.

Suivant la valeur de l'angle formé entre chaque entité, nous avons étudié les conditions de continuité du parcours d'usinage, afin de montrer les surfaces ne pouvant être usinées pour un diamètre d'outil donné.

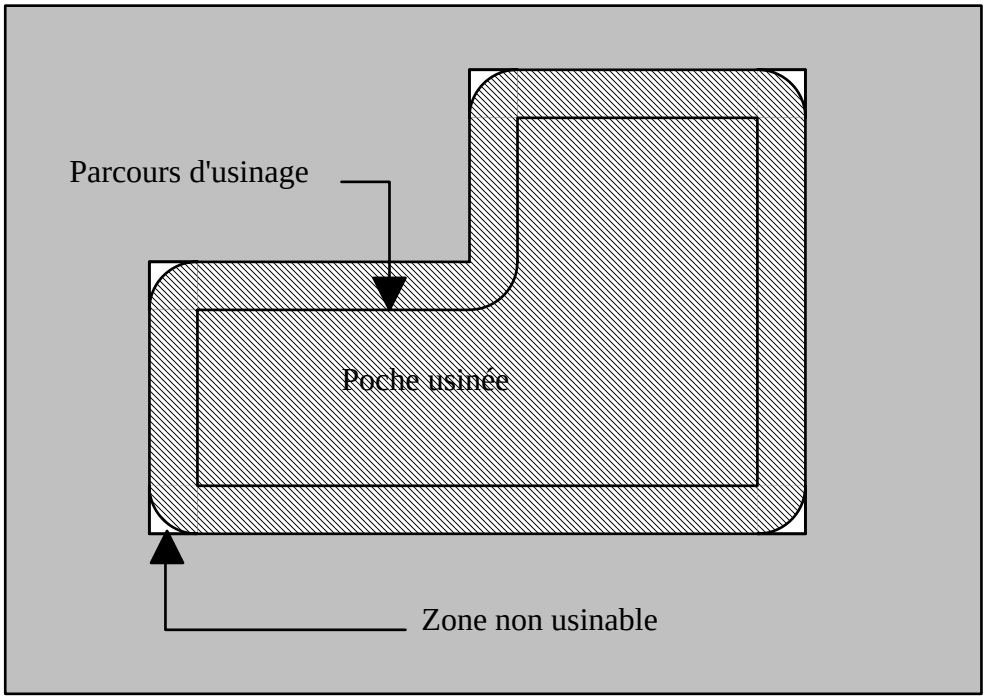


Figure 30 : Parcours d'usinage.

11.8. OPTIMISATION

Pour permettre une lecture plus rapide des résultats, nous avons choisi de calculer un rayon d'outil optimal. Celui-ci permet d'usiner la poche avec un rayon d'outil maximum, sans entraîner l'apparition d'interférences. Cette information permet de choisir l'outil pour réaliser le cycle de poche en ébauche, et ainsi de vérifier sa faisabilité.

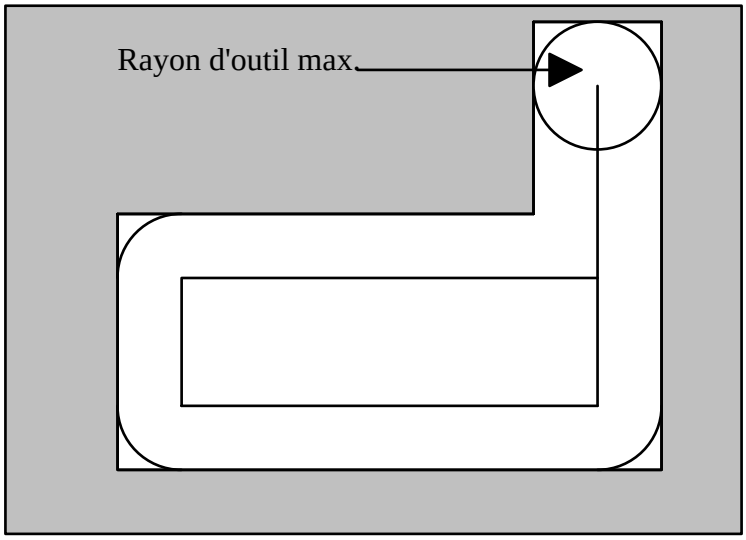


Figure 31 : Rayon d'outil max. sans îlot.

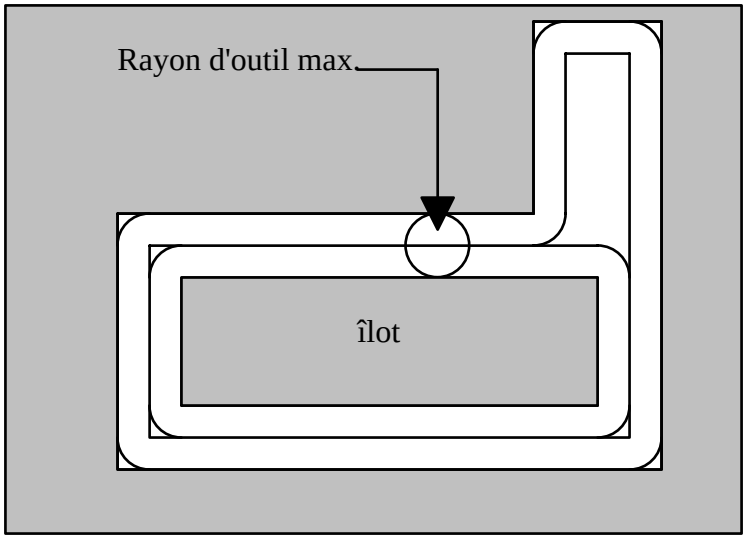


Figure 32 : Rayon d'outil max. avec îlot.

Pour calculer le rayon de l'outil, nous sommes partis d'un rayon quelconque et nous avons cherché si des interférences existent sur le parcours. Si elles n'existent pas, nous augmentons le rayon jusqu'à ce qu'elles apparaissent. Dans le cas contraire (présence d'interférences), nous diminuons le rayon pour les faire disparaître (§ 3. Algorithme de calcul du rayon d'outil).

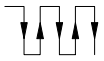
11.9. USINAGE D'UNE POCHE

Il existe deux types de stratégie pour réaliser une poche par fraisage :

- usinage par contours parallèles ;
- usinage par directions parallèles.

L'usinage par contours parallèles utilise les méthodes de décalage du ou des contours de la poche pour générer une spirale et que l'on dénomme l'« offset milling ».

L'usinage par directions parallèles a pour base une idée simple : choisir une ligne de référence. Les trajectoires d'usinage dérivent des segments parallèles, selon un pas donné ou optimisé. Deux possibilités se présentent alors :

- usiner en « zig-zag » ; 
- usiner en « zig ». 

Dans le cadre de notre recherche, nous avons choisi d'utiliser la stratégie en « zig ».

Pour réaliser l'usinage d'une poche, nous avons développé une série de fonctions permettant de calculer des intersections entre des segments ou entre des segments et des arcs. Pour chaque décalage de la trajectoire d'usinage, nous calculons toutes les intersections avec les entités constituant le contour de la poche et des îlots.

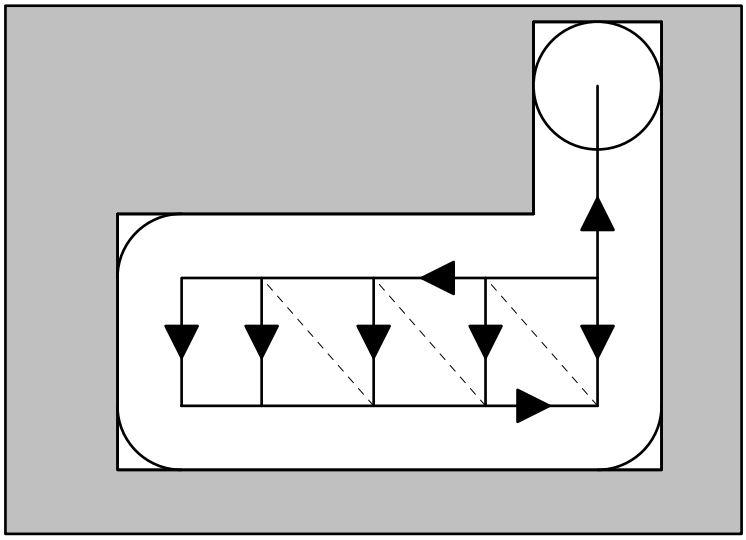


Figure 33 : Usinage de l'intérieur d'une poche.

11.10.CHOIX DE VISUALISATION ET RÉSULTATS

Nous avons choisi de visualiser les modifications à apporter au contour d'une poche sur une fenêtre graphique unique. Notre application permet de visualiser, l'une après l'autre, chaque poche créée dans Pro-Engineer en respectant les critères de forme décrits précédemment. Nous avons apporté une routine d'optimisation, qui permet de calculer le rayon de fraise maximum pouvant usiner la poche sans provoquer d'interférences entre îlot et poche ou entre les côtés de la poche. Pour améliorer la lecture du dessin, nous avons coloré la zone de la poche pouvant être usinée et ainsi mettre en relief les parties non usinées. Pour visualiser les zones provoquant des interférences, le rayon de la fraise peut être augmenté.

Nous avons restreint le nombre de commandes afin d'améliorer la lecture de notre interface. La structure de la fenêtre graphique est la suivante :

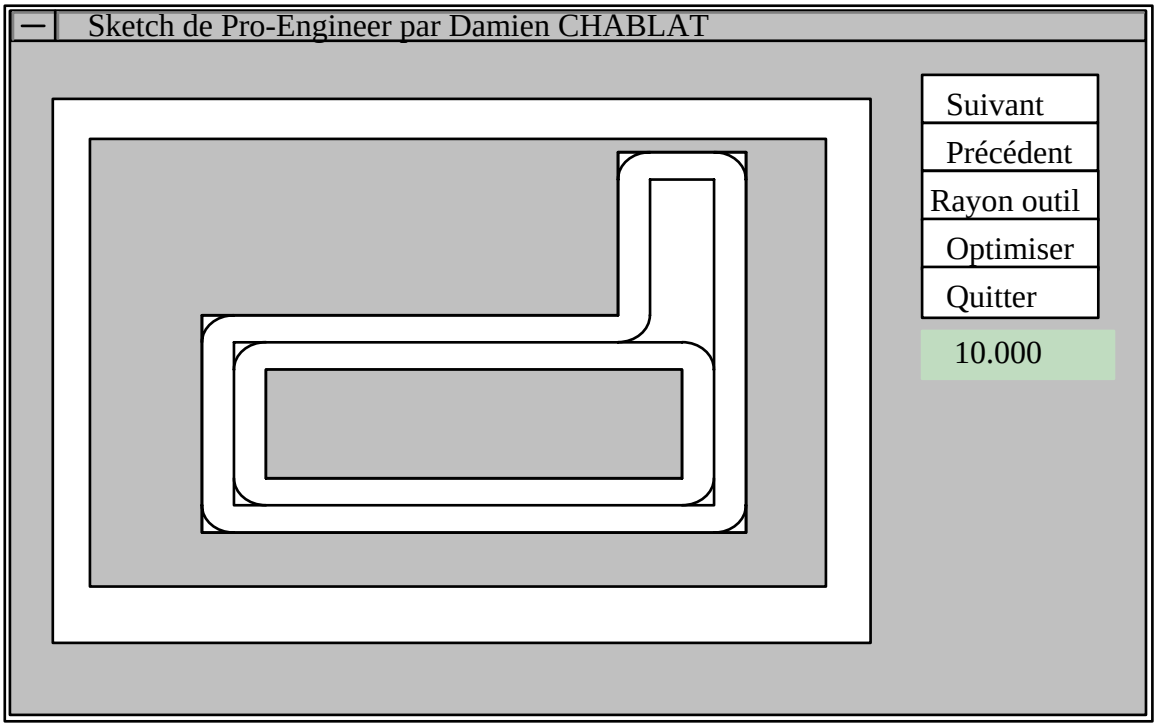


Figure 34 : Interface graphique de l'application.

12. TESTS DE VALIDATION

12.1. POCHE SIMPLE

Dans cet exemple, nous observons la présence d'une poche à l'intérieur de la poche. Le diamètre de fraise calculé est de 20 millimètres.

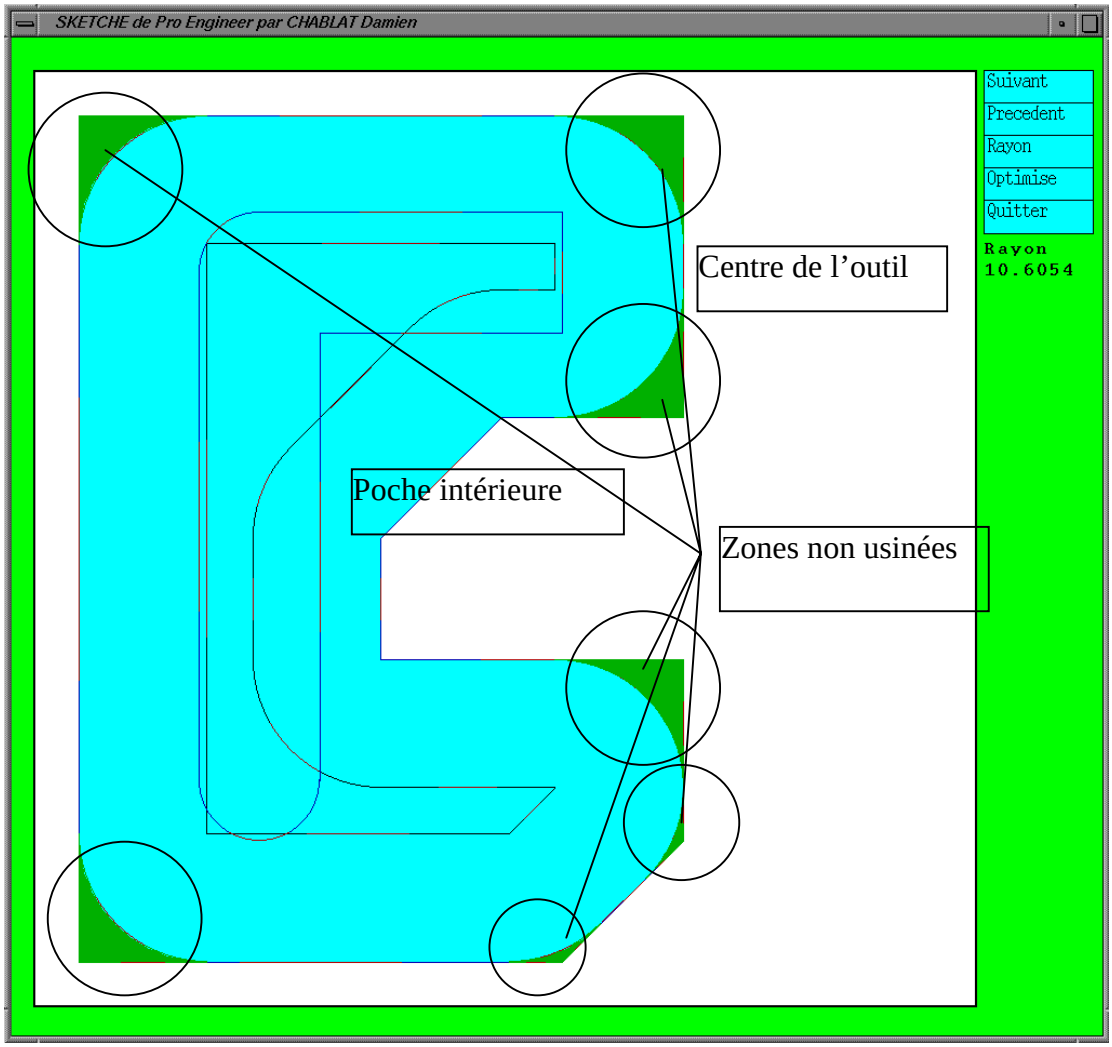


Figure 35 : Poche simple.

L'usinage de la poche a permis de vérifier la présence de surfaces non usinées et a montré l'importance que peut avoir ces « oublis » par rapport au volume de la poche.

12.2. POCHE AVEC ÎLOT

La présence d'un îlot entraîne une diminution du diamètre de l'outil pour permettre à celui-ci d'usiner entre l'îlot et le flan de la poche.

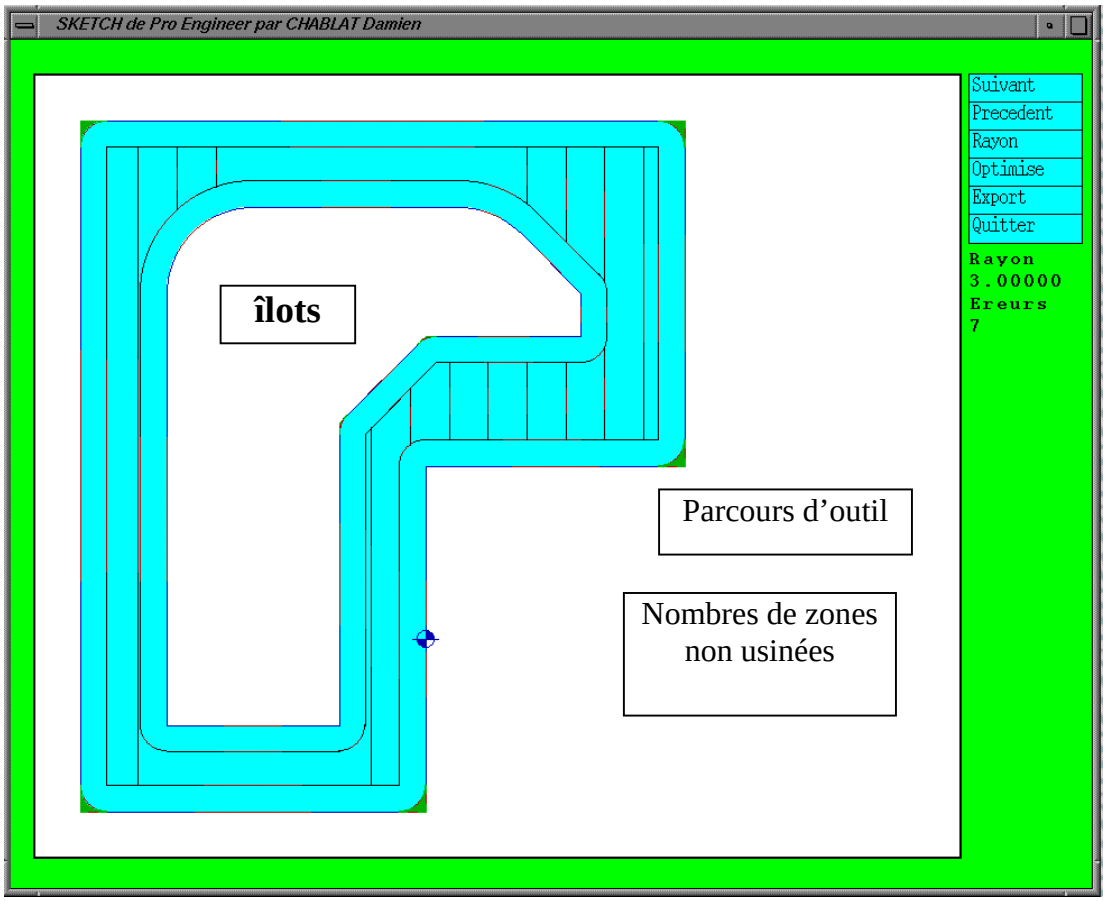


Figure 36 : Poche avec îlot.

L'utilisation de notre application permet à l'utilisateur de connaître le diamètre maximum pouvant usiner la poche sans entrer en collision avec l'îlot.

Dans un logiciel de F.A.O. traditionnel, il aurait choisi son outil et il s'en serait aperçu après avoir exécuté une simulation d'usinage. Ces erreurs auraient obligé l'utilisateur à reprendre le modèle C.A.O. pour modifier le modèle et le rendre identique à la forme de la pièce usinée.

12.3. MODIFICATION DU RAYON D'OUTIL

L'application permet à l'utilisateur de modifier le rayon de l'outil.

Figure 37 : Modification du rayon de l'outil.

12.4. CONTRIBUTIONS A L'INTÉGRATION C.A.O. / F.A.O.

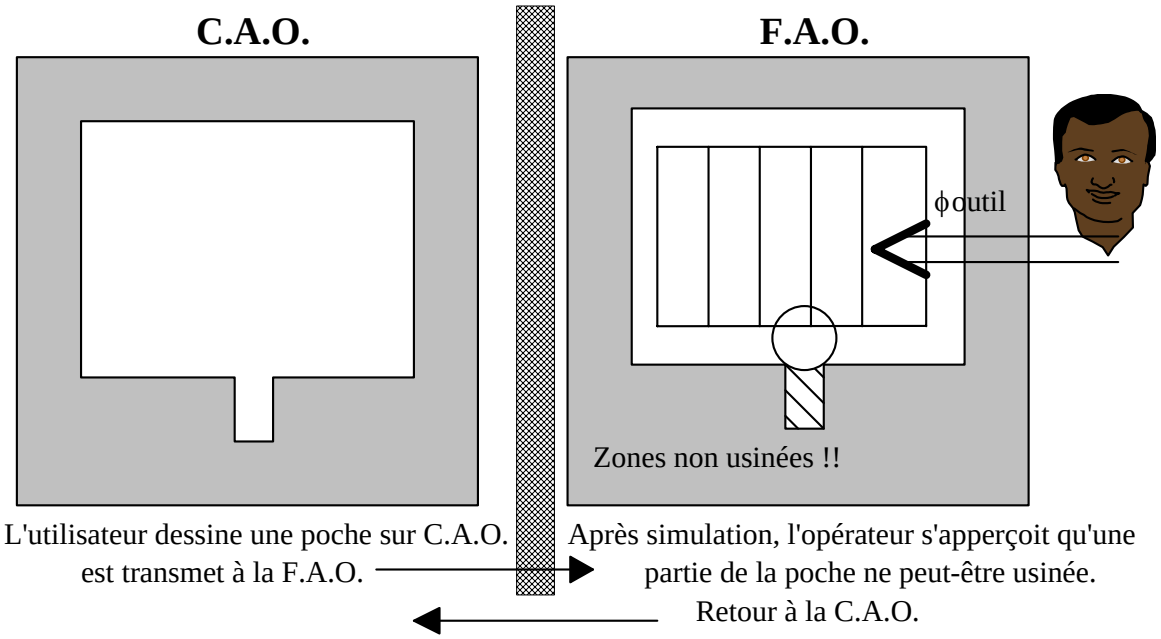


Figure 38 : Dialogue actuel entre C.A.O. / F.A.O.

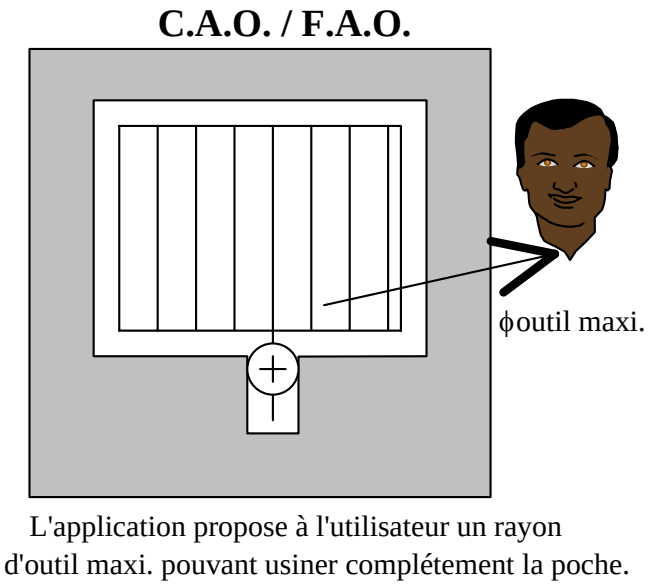


Figure 39 : Contributions a l'intégration C.A.O. / F.A.O.

13. CONCLUSION

L'introduction de contraintes de fabrication dans un modeleur géométrique est possible dans le cas de la réalisation de la fonction poche. Cependant, pour améliorer le niveau d'intégration, nous devrions mieux exploiter les informations topologiques présentes dans le modeleur. Les fonctions offertes aux développeurs par Pro-Develop ne permettent pas actuellement d'accéder à toutes les fonctions inclues dans Pro-Engineer.

Pour la fabrication, notre application rompt avec les logiciels de F.A.O. actuels dans lesquels les utilisateurs définissent leurs outils et les surfaces à usiner et ensuite réalisent l'usinage. Dans notre cas, l'utilisateur sélectionne une entité, la poche, et le système lui propose un outil en lui visualisant les surfaces qui ne pourront pas être usinées. Ce système, couplé à une base de données d'outils, pourrait permettre, au dessinateur, d'utiliser des dimensions compatibles avec les outils de fabrication.

TROISIÈME PARTIE :

ANNEXES

14. PRO-DEVELOP

14.1. LES FEATURES

Les *Features* sont identifiées dans *Pro-Develop* en utilisant un entier. C'est le même identificateur qui apparaît dans *Internal Feature* ID quand on utilise la fonction *Model Info* ou *Feature Info* de *Pro-Engineer*.

La fonction prodb_get_feature_ids() donne la liste des identificateurs de Features comprise dans le modèle. Il est possible de filtrer les Features suivant plusieurs propriétés. Par exemple, il est possible d'inclure ou d'exclure les Features effacées.

int prodb_get_feature_ids(p_model, filter, feature_ids)

*char *p_model;*

int filter;

*int **feature_ids;*

Cette fonction nécessite l'appel de *pro_get_current_object* afin de connaître l'adresse de l'objet en cours. Le résultat de cette fonction est un tableau contenant les informations sur chacune des Features.

Pour connaître leurs attributs, nous devons utiliser les fonctions *prodb_feature_info* et *prodb_get_feature_type*.

prodb_get_feat_type(p_model, feat_id)

*char *p_model;*

int feat_id;

Cette fonction donne le type de la Feature : protrusion, poche, découpe, etc..

prodb_feature_info(p_model, feat_id, feat_info)

*char *p_model;*

int feat_id;

*Pro_feature *feat_info;*

Cette fonction donne une variable de type structure, dans laquelle se trouve tous les attributs d'une Feature.

Attention : Avant d'utiliser cette fonction, il est nécessaire d'allouer de la mémoire pour la structure *Pro_feature.*

feat_info = (Pro_feature) malloc(sizeof(Pro_feature))*

14.2. LES MENUS

Les fichiers MENUS :

Les fichiers menus permettent de spécifier les textes pour chaque commande et l'aide en ligne qui apparaît lorsque l'on place la souris sur le texte de la commande.

Pro-Engineer cherche les fichiers menus de Pro-Develop dans les répertoires suivants :

- Le répertoire de recherche en cours ;

- Le répertoire *text/menus* dont le chemin est enregistré dans le fichier *Prodev.dat* par la commande *TEXT_PATH*.

Ce placement est vivement recommandé par PTC.

Nom et contenu des fichiers menus :

Il existe deux extensions pour les fichiers menus :

- *.mnu* Ce sont des fichiers qui décrivent des menus complets ;

- *.aux* Ce sont des fichiers qui décrivent de nouvelles commandes qui s'additionnent aux menus existants de Pro-Engineer.

Les restrictions suivantes sont appliquées aux noms des fichiers :

- Le nom ne doit pas avoir plus de 14 caractères ;

- Le nom doit être unique dans tout Pro-Engineer.

Tous les menus utilisés par Pro-Engineer se trouvent dans le sous-répertoire de Pro-Engineer, soit actuellement *v14/text/usascii/menus* et la version française sous le sous-répertoire */v14/text/french/menus*.

Pour la création de commandes additionnelles aux menus existant, il est préférable d'utiliser le même nom que Pro-Engineer avec l'extension *.aux*.

Syntaxe et sémantique des fichiers menus :

Les deux types de fichiers *.mnu* et *.aux* ont le même format.

Le format consiste en un groupe de 3 lignes, un groupe pour chaque commande d'un menu, et un groupe au début du fichier pour le titre. Ce groupe contient le titre du menu sur la première ligne, et deux lignes blanches.

Le titre du menu est le nom qui apparaît sur l'en-tête du menu quand on utilise Pro-Engineer. Le titre des menus se rapporte aux codes de Pro-Develop, et il est essentiel que ce nom soit identique dans tous les menus de Pro-Engineer. Par exemple, si l'on écrit un fichier *.aux* pour ajouter des commandes aux menus de Pro-Engineer, on doit s'assurer que ce titre est identique au fichier correspondant *.mnu*. Si l'on crée un nouveau menu, on doit aussi s'assurer que ce nom n'est jamais utilisé par Pro-Engineer.

Chaque groupe de commande comprend les 3 lignes suivantes :

- Nom de la commande. Si le nom qui apparaît sur l'écran de Pro-Engineer contient des espaces, chaque espace doit être remplacé par le caractère # dans les fichiers menus. Si le nom de commande est suivi par un autre nom, séparé par un

espace, le second est celui qui apparaît à l'écran. Le premier nom est encore utilisé pour faire référence au code de Pro-Develop.

- L'aide en ligne. Cette seule ligne de texte explique les commandes.

- Autre aide. Si cette ligne n'est pas blanche (ou ne commence pas par le caractère #), elle est utilisée à la place de l'aide en ligne. Il apporte une alternative à l'aide en ligne.

Ajouter de nouvelles commandes dans les menus :

- *promenu_create() ;*

- *promenu_expand() ;*

- *promenu_on_button().*

Lorsque l'on ajoute une nouvelle commande dans un menu existant, on modifie la définition des menus de Pro-Engineer en mémoire.

Pour ajouter des commandes dans un menu, écrire le fichier menu et l'ajouter dans la fonction *user_initialize()*, les étapes sont :

- Lecture par Pro-Engineer en mémoire, en utilisant *promenu_create()* ;

- Ajouter les commandes dans votre fichier de menu, en utilisant *promenu_expand() ;*

- Définir les actions de ces nouvelles commandes, en utilisant *promenu_on_button().*

promenu_create() :

Les arguments de *promenu_create()* sont :

- *char *menuname.* Le titre unique du menu qui apparaît sur la première ligne du fichier menu et qui apparaît sur l'en-tête du menu à l'écran.

- *char *filename*. Le nom du fichier de menu, incluant l'extension du fichier mais pas le chemin d'accès (répertoire).

La fonction renvoie un identificateur du menu de type entier, dont l'on a normalement besoin. Si une erreur apparaît lors de l'exécution de la fonction, la fonction renvoie la valeur -1. Si la fonction est rappelée une seconde fois, elle n'a aucun effet.

promenu_expand() :

Cette fonction a les mêmes arguments et retourne les mêmes valeurs que la fonction *promenu_create()*. A la place de charger en mémoire un nouveau menu, la fonction rajoute de nouvelles fonctions dans le menu existant en mémoire.

promenu_on_button() :

Les trois premiers arguments de *promenu_on_button()* sont les suivants :

- *char *menuname*. Le titre du menu contenant la fonction.

- *char *itemname*. Le premier nom de la fonction dans le fichier de commande, et non le second, mais avec des espaces à la place du caractère #.

- *int (*function)()*. Un pointeur pour appeler une fonction de Pro-Develop quand l'utilisateur sélectionne une commande dans le menu. Pour passer un pointeur sur une fonction, il faut mettre le nom de la fonction sans les parenthèses. Si la fonction n'a pas déjà été appelée dans le même fichier, il est nécessaire d'ajouter une déclaration pour que le compilateur sache que c'est une fonction et non une variable de type entier.

Les deux autres arguments *arg1* et *arg2* sont des arguments optionnels pour la fonction. Ces arguments permettent à la fonction d'être plus flexible. Si ces paramètres ne sont pas nécessaire, il faut les remplacer par la valeur *NULL* et *0*.

Nouveau menus

Les fonctions suivantes sont introduites :

- *promenu_action()* ; - *promenu_make()* ;

- *promenu_exit_up()* ; - *promenu_no_exit()*.

Pro-Develop permet de créer de nouveaux menus. La définition de nouveaux menus diffère de l'adjonction de nouvelles fonctions dans des menus excitants sur les points suivants :

- Le nom du fichier de menu a pour extension *.mnu* et nom *.aux*, mais ils ont la même syntaxe ;

- Il n'est pas nécessaire d'appeler la fonction *promenu_expand()* car toutes les fonctions sont écrites dans le même fichier.

- Il est nécessaire de définir une fonction pour sortir du menu, et ajouter une action pour chaque commande du menu ;

- Il faut aussi déclarer ce nouveau menu dans la fonction *user_initailize()* ou localement dans une fonction, juste avant de l'utiliser.

Exit action :

Il est nécessaire de faire connaître au gestionnaire de menu de Pro-Engineer que chaque fonction doit être appelée pour chaque commande du menu, et ces fonctions sont appelées si l'on sélectionne une fonction dans un autre menu.

Note : Si aucune fonction de sortie n'est définie, le programme entre en collision, si l'utilisateur sélectionne une commande dans un autre menu.

Il y deux types de fonctions permettant de sortir d'un menu :

- *Nothing*. Le menu sélectionné est ignoré. Ceci est utilisé si l'on désire que l'utilisateur sélectionne plusieurs actions avant de quitter le menu courant.

- *Close the current menu*. Les menus "*unwind*" pour le niveau de commande sélec*tionné,* et que la commande sélectionnée commence. C'est la fonction habituelle pour quitter un menu.

Certaines commandes de Pro-Engineer peuvent être exécutées sans quitter le menu courant, tel que la fonction *view* et *Misc*. Il est possible d'exécuter ces commandes alors que d'autres commandes sont en court.

Comment définir un nouveau menu :

Pour définir un nouveau, il faut commencer par écrire un fichier de menu. Pour pouvoir utiliser ce nouveau menu, il faut ajouter les commandes suivantes dans le programme :

- Lecture en mémoire du menu par Pro-Engineer, en utilisant la fonction *promenu_create()* ;

- Définir les nouvelles fonctions associées aux menus en utilisant la fonction *promenu_on_button()* ;

- Définir une commande pour sortir du nouveau menu en utilisant la fonction *promenu_on_button()* et une fonction de sortie pour menus.

Pour définir une action de sortie d'un menu, il faut rappeler une seconde fois la fonction *promenu_on_button()*, mais à la place du nom de la fonction (second argument), il faut indiquer le nom du menu. Si vous voulez que le menu se déroule, et qu'une nouvelle fonction puisse être validée, utilisez la fonction *promenu_exit_up()*.

Si vous désirez, que la sélection d'une nouvelle fonction (à l'extérieur du menu en cours), soit ignorée, utilisez la fonction *promenu_exit_up()*. Si vous vous servez de cette fonction, il est nécessaire de définir une fonction pour pouvoir sortir de ce menu. Par exemple, il est possible de définir une commande spécifique (Done),

permettant de sortir de ce menu, et utiliser comme commande la fonction *promenu_exit_up()*.

Utiliser un nouveau menu :

Après avoir défini un nouveau menu, il faut savoir comment l'utiliser. Ce menu est normalement appelé à l'intérieur d'une commande d'un autre menu. Pour cela, il faut exécuter les fonctions suivantes :

- Afficher le menu à l'écran, en utilisant la fonction *promenu_make()* ;

- Rendre actif ce menu, pour que l'utilisateur puisse le sélectionner, en utilisant la fonction *promenu_action()*.

Appel de la fonction promenu_make :

Le premier argument de la fonction *promenu_make()* est le nom du menu principal ou du sous-menu. Le nom habituel est *MAIN_MENU*. Le second argument est le titre du menu.

Appel de la fonction promenu_action :

La fonction *promenu_action* n'a qu'un seul argument, le titre du menu. Si ce titre est le nom du dernier menu appelé, il peut être remplacé par une chaîne de caractères vide *""*. La valeur que renvoie cette fonction est importante, si on utilise la fonction *promenu_exit_action_up()* comme fonction pour sortir de ce menu.

La fonction *promenu_action()* se termine seulement lorsque le menu est fermé, et que la fonction *promenu_exit_up()* ou *promenu_exit_action_up()()* a été appelée. Le programme peut continuer après l'appel de la fonction *promenu_action* que si le menu est fermé.

15. X11

15.1. LES BUTS DE X

- Utilisation sur une grande variété de dispositifs d'affichage ;
- Indépendance vis à vis du terminal ;
- Utilisation en réseaux ;
- Plusieurs applications sur le même terminal ;
- Souplesse ;
- Fenêtrage ;
- Graphiques bi-dimensionnels ;
- Image ;
- Extensions.

La norme X11 définie un environnement graphique multitâches, multi-utilisateurs et permet le multi-fenêtrage.

15.2. ECRAN ET AFFICHAGE

Chaque application est liée à un DISPLAY[9], celui-ci est composé d'un clavier, d'un pointeur (souris, boule...), et d'un ou plusieurs écrans.

15.3. MODÈLE CLIENT-SERVEUR

Par serveur, il faut entendre serveur graphique.

- X est destinée à être utilisé en réseau ;
- Une application peut être exécutée sur un système différent de celui effectuant l'affichage ;

- Protocoles TCP/IP[10] et DECNET.

Le serveur X :

- Gère l'accès des clients à l'afficheur ;
- Interprète les messages réseaux ;
- Envoie les messages aux clients ;
- Maintient les ressources ;
  - fenêtres ;
  - curseurs ;
  - fontes ;
  - contextes graphiques ;

  .....

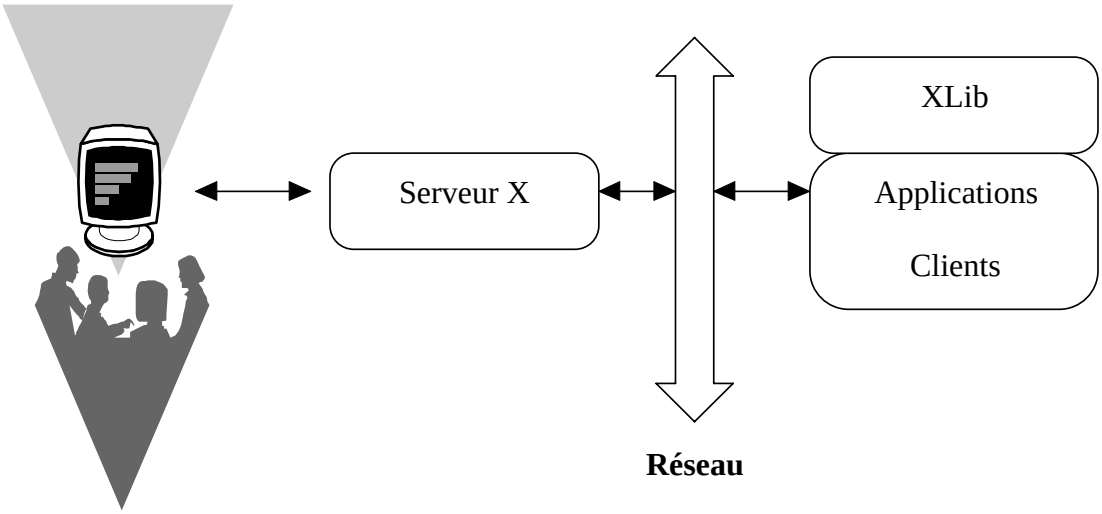


Figure 40 : Modèle Client-Serveur.

[9] Display : Ecran.

[10] TCP/IP : Protocole de transfert Inthernet.

15.4. LES ÉVÉNEMENTS

Un client doit pouvoir réagir à des événements asynchrones :

- *Button Press*[11], *Button Release*[12] ou *Motion Notify*[13] pour les événements de la souris ;
- *Key Press*[14] et *Key Release*[15] pour les actions du clavier ;
- *Expose* pour toutes les modifications de position de fenêtre ;
- *Configure Notify*[16] pour les modifications de la grandeur des fenêtres.

Les événements sont chaînés dans une file d'attente. C'est le client qui sélectionne les événements.

Les difficultés principales de ce type d'environnement sont la gestion des événements. Ceux-ci doivent être filtrés et gérés séparément selon leur nature et leur origine.

Dans un programme pour PC[17] ou pour MAC[18], le programmeur doit gérer les entrées (clavier, disque, pointeur de souris) et les sorties (écran, disque et imprimante). Sous X11, l'écran est composé de fenêtres ayant des caractéristiques propres. Le but est de distribuer les événements leur étant attachés et ne pas perturber le reste du système.

---

[11] Button Press : Appuyer sur le bouton de la souris.
[12] Button Release : Relache le bouton de la souris.
[13] Motion Notify : Modification de la position de la souris.
[14] Key Press : Presser une touche du clavier.
[15] Key Release : Annulation de la pression sur la touche du clavier.
[16] Configure Notify : Modification de la configuration.
[17] PC : Personal Computer
[18] MAC : Macintosh.

15.5. LES CLIENTS.

Les applications :

- Elles communiquent avec le serveur par des appels à Xlib ;
- La Xlib est une bibliothèque C ;
- Les appels à la bibliothèque Xlib sont traduits en requêtes et envoyés au serveur X ;
- Le protocole de communication est le protocole X.

15.6. LES TOOLKITS.

Ce sont des bibliothèques au-dessus de la bibliothèque Xlib, ils complètent des interfaces utilisateurs.

- Menus ;
- Boutons ;
- Widget[19].

Exemples de Toolkits[20] :

- Xt (MIT[21] & DEC[22]) ;
- Xm (OSF[23] / Motif) ;
- Sun (Open lock) ;
- Andrew.

L'utilisation de ces bibliothèques compromet la portabilité d'un programme.

[19] Widget : Fenêtre.

[20] ToolKits : Boîte à outils.

[21] M.I.T. : Massachussetts Institue of Technology.

[22] D.E.C. : Digital Equipement Corporation.

[23] O.S.F. : Open Software Foundation, Inc

15.7. COMPILATION ET LIBRAIRIE.

La compilation du programme se fera grâce à un Makefile[24]. Dans ce fichier, il est possible de modifier le chemin d'accès aux librairies. Le programme doit pouvoir trouver les Headers[25] associés à Xlib. Sur Silicon Graphic, ces informations sont présentes sous le répertoire */user/include.*

15.8. EXEMPLE

L'exemple qui suit, nécessite la présence des headers suivants :

```
#include <X11/Xlib.h>
#include <X11/X.h>
#include <X11/Xutil.h>
```

Initialisation

La création d'une application Xlib commence toujours par la création d'une fenêtre graphique. Cette opération se décompose en trois étapes :

```
display = XOpenDisplay(display_name));
screen = Default(display);
```

[24] Makefile : Script de compilation.

[25] Header : En-tête de programme.

```
window = XCreateSimpleWindow(display,
                             RootWindow(display, screen),
                             window_x,
                             window_y,
                             window_largeur,
                             window_hauteur,
                             bordure,
                             BlackPixel(display, screen),
                             WhitePixel(display, screen));
```

Ensuite, il est nécessaire de créer un contexte graphique, les propriétés de notre fenêtre graphique et de filtrer les entrées standards.

```
window_gc = XCreateGC(display,
                      window, 0, 0);
XSetStandardProprieties(display,
                        window,
                        window_name,
                        "Demo",
                        None,
                        argv,
                        argc,
                        1size_hints);
XSelectInput(display,
             window,
             ExposureMask | keyPressMask |
             PointerMotionMask | StructureNotifyMask);
```

Toutes ces fonctions ont permis d'initialiser notre application, il ne reste plus que l'affichage.

XMapRaised(display, window);

Dans tout cet exemple, nous avons utilisé des variables dont la syntaxe est la suivante :

Dsiplay	*display;*
int	*screen;*
Window	*window;*
XSizeHints	*size_hints;*
GC	*window_gc;*
unsigned int	*window_x,*
	window_y,
	window_largeur,
	window_hauteur;

16. ALGORITHME DE CALCUL DU RAYON D'OUTIL

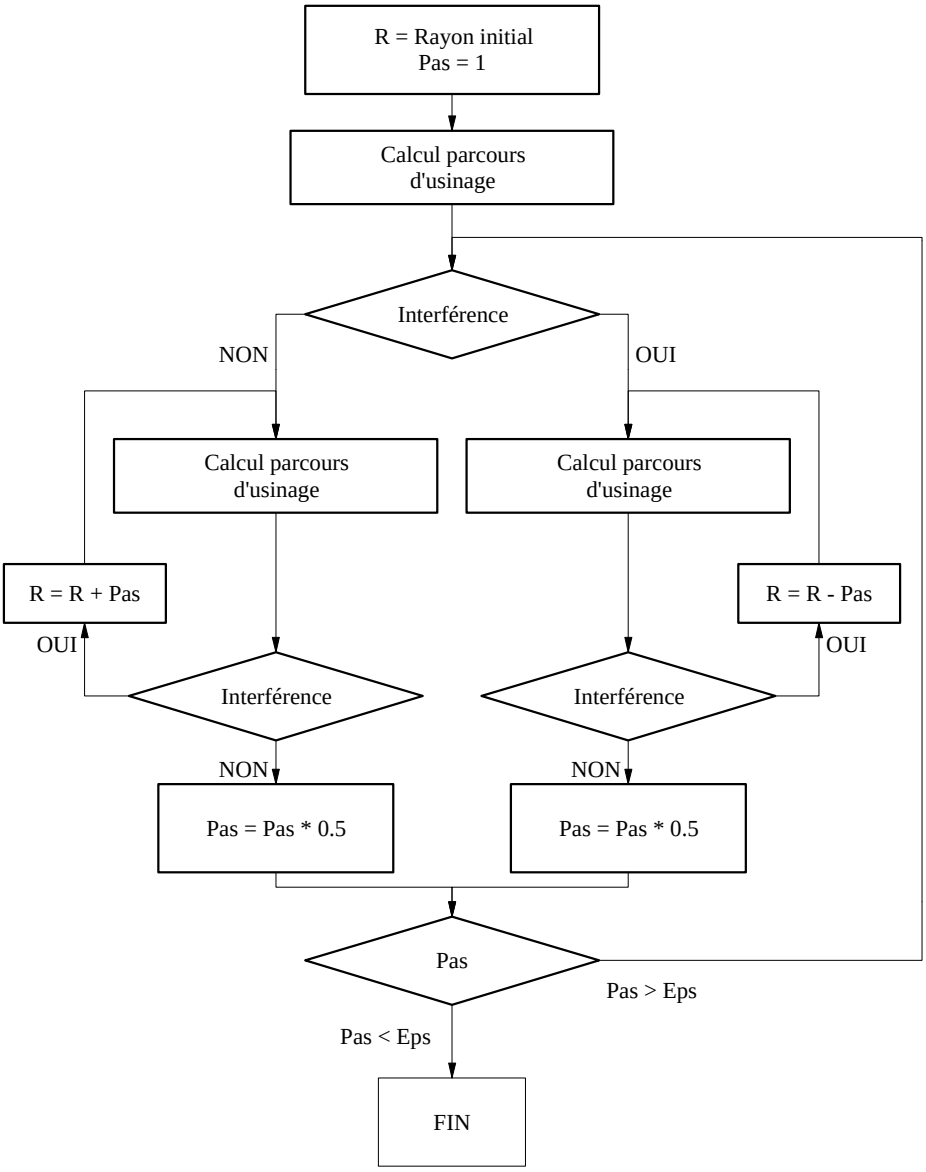


Figure 41 : Algorithme de calcul de rayon d'outil.

17. LANGAGE DE PROGRAMMATION POUR COMMANDES NUMÉRIQUES

Pour valider notre travail, nous avons développé une fonction permettant de créer des fichiers pour commandes numériques. Ainsi, nous avons pu constater les surfaces non usinées de la poche.

Notre programme fonctionne sur un centre d'usinage à commande numérique de type *MIKRON WF 41 C*, et les fonctions utilisées sont les suivantes :

- Déplacement rapide G0 ;
- Interpolation linéaire G1 ;
- Interpolations circulaires G2 et G3.

18. STRUCTURE DE L'APPLICATION

18.1. STRUCTURE GLOBALE

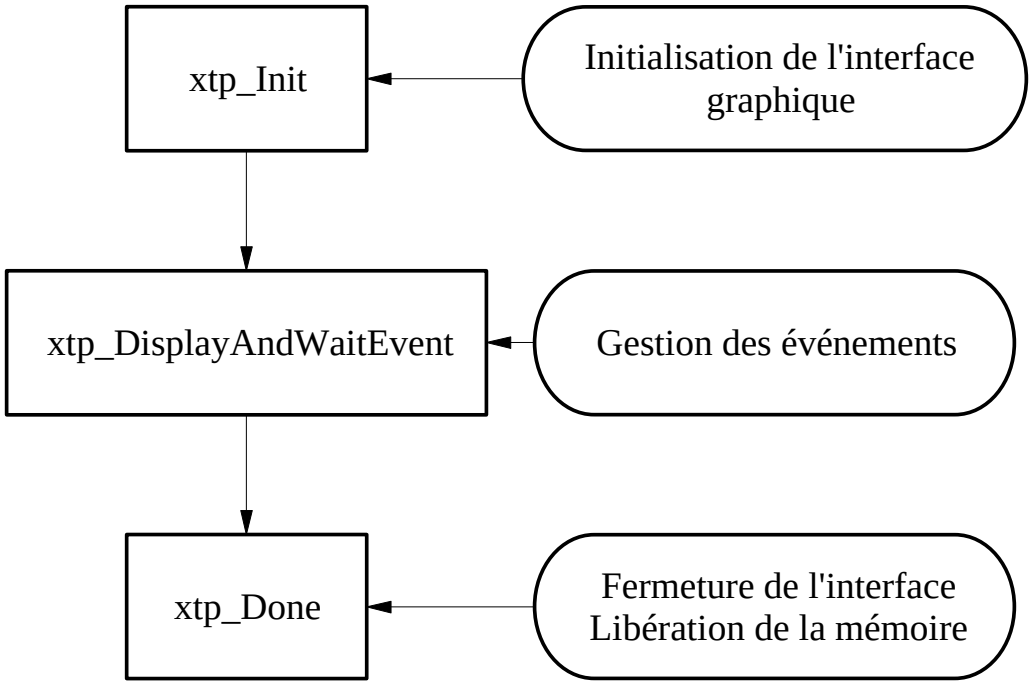


Figure 42 : Structure globale.

Notre application fonctionne en gérant les *événements* provenant de l'interface X. Pour permettre une relecture de notre programme, nous avons détaillé, dans le paragraphe suivant, les différentes fonctions utilisées dans *wtp_DisplayAndWaitEvent.*

18.2. GESTION DES ÉVÉNEMENTS

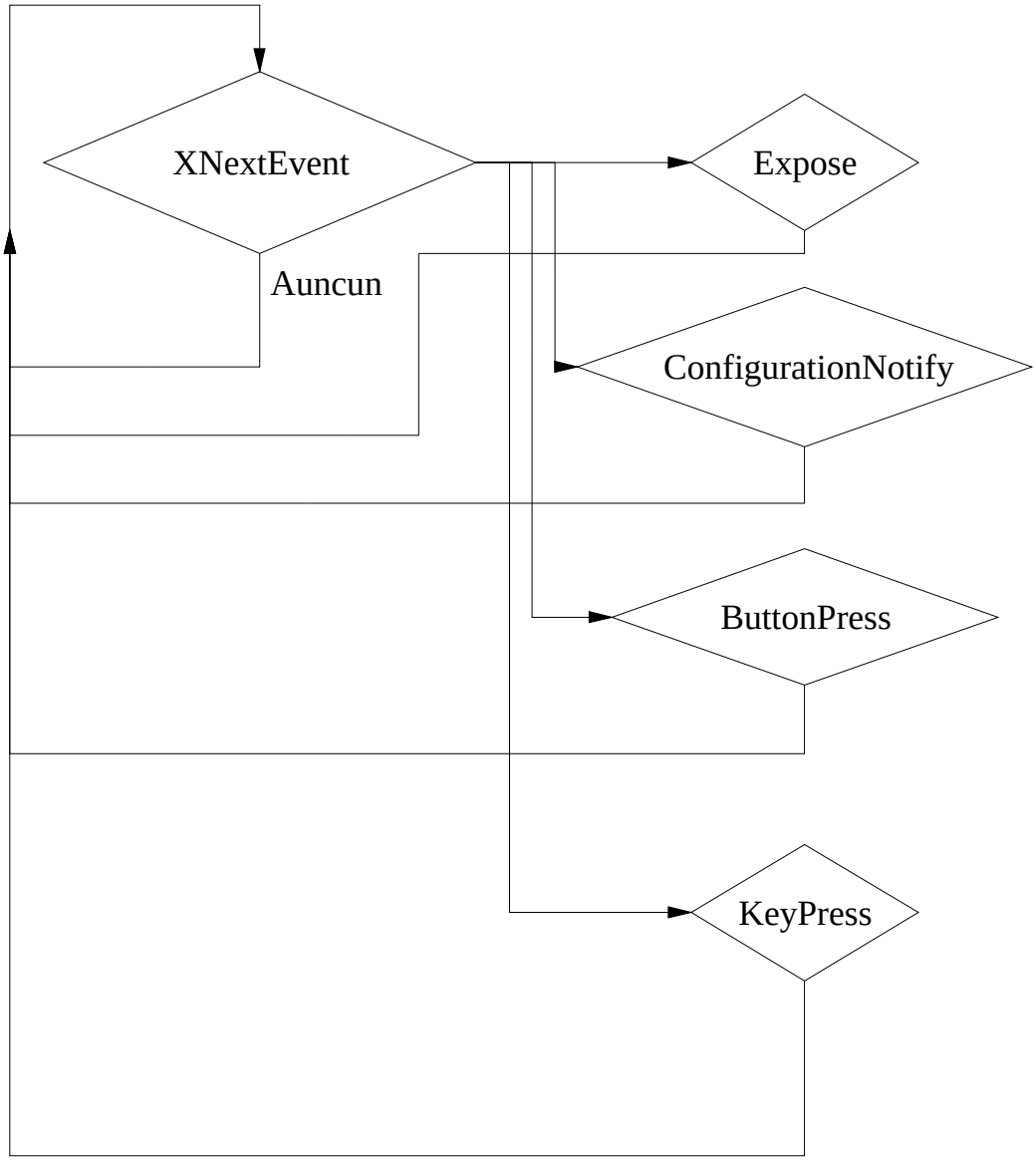


Figure 43 : Gestion des événements.

19. RÉFÉRENCES BIBLIOGRAPHIQUES

Schulz Herbert ; Schützer Klauss ; FINDES - Integrating Design and Manufacturing ; Valencienne 1994 ; Pages 43 - 57.

Schulte Richard M. ; Padmanabhan Srikanth ; Devgun Mohan S. ; Feature - driven, process - based approach to the integration of CAD / CAM in wireframe models ; Internationnale Journal of Production ; Volume 30 ; Number 5 ; 1992 ; Pages 1005 - 1028.

Houten Fred van ; Erve Tom van't ; PART, a Feature Based Computer Aided Process Planing system.; Revue internationnale de CFAO et d'infographie, Volume 7 ; Number 3 ; 1992 ; Pages 335 - 368.

Gardan Yvon ; Minich Christian ; La modélisation géométrique et l'extraction de caractéristiques de forme.; Revue internationnale de CFAO et d'infographie, Volume 7, Number 3 1992 ; Pages 311 - 333.

Mawussi K. ; Bernard A. ; Bourdet P. ; Feature modelling and machining of forming dies.; Valencienne 1994 ; Pages 525 - 530.

Krause F. L. ; Rieger E. ; Ulbrich A. ; Feature Processing as Kernel for Integrated CAE systems.; Valencienne 1994 ; Pages 693 - 716.

Tönshoff H. K. ; Aurich J. C. ; Baum Th. ; Configurable Feature based CAD / C.A.P.P. System ; Valencienne 1994 ; Pages 757 - 767.

De Martino T. ; Falcidieno B. ; Giannini F. ; Hassinger S. ; Ovtccharova J. ; Feature Based modelling by integrating design and recognition approches.; Computer-Aided Design Volume 26 Number 8 August 1994 ; Pages 646-653.

Schulte Michael ; Stack Rainer ; Weber Christian ; Feature Structure for the Representation of Design Objects as Combinations of Technical Functions and Geometry in CAD ;

Fields M. C. ; Anderson D. C. ; Fast Feature extraction for machining applications.; Computer Aided Design ; Volume 26 ; Number 11 ; November 1994 ; pages 803 - 813.

Delbressine F. L. M. ; Hijink J. A. W. ; Discrete Part Design by Taking Manufacturing Restrictions into Account.; Annals of the CIRP ; Volume 40 / 1 / 1991 ; pages 171 - 174

Corbett J. ; Woodward J. A. ; Delbressine F. L. M. ; **Hijink J. A. W.** ; Annals of the CIRP, Volume 40 / 1 / 1991

De Jong Frans ; Reitsema Tjeert ; Hoogeboom Rick ; Feature Based Modelling a way ti improve the CAD / CAM process.; Revue internationnale de CFAO et d'infographie. Volume 7 - n 2 1992 pages 209-234

Shah Jami J. ; Rogers Mary T.; Functional requirements and conceptual design of the Feature Based Modelling System ; Computer Aided Engineering Journal ; February 1988 ; pages 9 - 15

Schulz Herbert ; Schützer Klauss ; FINDES - Integrating Design and Manufacturing.; Valencienne 1994 ; pages 43 - 57

Perotti G. ; Tornincasa S. ; Oberto G. ; Semantic Tecniques for Representation and Identification of Part Families.; Annals of the CIRP ; Volume 40 / 1 /1991 ; pages 451 - 545

Mony Charles ; Un modèle d'intégration des fonctions Conception - Fabrication dans l'ingénieri de produit; Thèse de doctorat ; Ecole centrale Paris ; 1992

Parametric Technology Corporation ; Pro-Develp User Guide ; 1994.

Parametric Technology Corporation ; Pro-Develp Reference Guide 1994.

Delannoy Claude ; Programmer en langage C ; Eyrolles ; 1993.

Nye Adrian ; Xlib Programming Manual ; Volume One ; O'Reilly & Associates, Inc.